Development of Problem Solving Environment For Handling Middleware Faults in Grid

By

Yogesh R.Vaghela

09MCE017

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERISITY

AHMEDABAD-382481

May, 2012

Development of Problem Solving Environment For Handling Middleware Faults in Grid

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Yogesh R.Vaghela

(09MCE017)

Guided By

Dr.Madhuri Bhavsar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERISITY

AHMEDABAD-382481

May, 2012

Declaration

I, **Yogesh R.Vaghela**, **09MCE017**, give undertaking that the Major Project entitled **"Development of Problem Solving Environment For Handling Middleware Faults in Grid"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Institute of Technology of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Yogesh R.Vaghela

Certificate

This is to certify that the Major Project entitled "Development of Problem Solving Environment For Handling Middleware Faults in Grid" submitted by Yogesh R.Vaghela (09MCE017), towards the fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr.Madhuri Bhavsar
Guide, Associate Professor,
Department of C.S.E.,
Institute of Technology,
Nirma University, Ahmedabad.

Dr.S.N.Pradhan
Professor and PG-Coordinator,
Department of C.S.E,
Institute of Technology,
Nirma University, Ahmedabad.

Prof.D.J.Patel
Professor and Head,
Department of C.S.E,
Institute of Technology,
Nirma University, Ahmedabad.

Dr.K.Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad.

Abstract

Grid enable the sharing, selection and aggregation of a wide variety of resources including supercomputers, storage systems, data sources and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering and commerce.However, the major challenge in such highly heterogeneous and complex computing environment is to design an efficient problem solving environment.The large number of components and their distribution in grid as well as the dynamic structure of the grid increases the risk of failures.Thus providing a scalable and generic problem solving environment as a basic service for grids, is of primary importance.This thesis aims at furnishing fault tolerant grid system by identifying and handling middleware faults.Grid middleware is core component of grid system responsible for overall progress of grid application.In the research work carried out during this thesis, middleware undertaken for experimentation is alchemi.Alchemi is a .NET-based grid computing framework that provides the runtime machinery and programming environment required to construct computational grid.Alchemi supports scheduling algorithm which selects highest priority thread among the pool of threads and dedicate to it to the available executor keeping scope for enhancement of application performance.This thesis proposes customized scheduler algorithm for handling faults so that scheduling of jobs will be based on available CPU power.Also a new manager proposed, recovers the jobs from its failure, providing problem solving environment for grid application resolving faults and its impacts.

Acknowledgements

My deepest thanks to **Dr.Madhuri Bhavsar**, Senior Associate Professor, Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad the Guide of the project that I undertook for giving her valuable inputs and correcting various documents of mine with attention and care. She has taken the pain to go through the project and make necessary amendments as and when needed.

My deep sense of gratitude to **Dr.S.N.Pradhan**, Professor and PG-Coordinator of Department of Computer Engineering, Institute of Technology, Nirma University, Ahmedabad for an exceptional support and continual encouragement throughout part one of the Major project.

I would like to thanks **Dr.Ketan Kotecha,** Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for his unmentionable support, providing basic infrastructure and healthy research environment.

I would also thank my Institution, all my faculty members in Department of Computer Science and my colleagues without whom this project would have been a distant reality. Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete my dissertation work successfully.

- Yogesh R.Vaghela
09MCE017

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Grid Computing Overview

Grid [1][2] is a type of distributed system that supports the sharing and coordinated use of geographically distributed and multi- owner resources independently from their physical type and location in dynamic virtual organizations that share the same goal of solving large-scale applications.
Rajkumar Buyya defined the Grid as:

"Grid[3] is type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed resources dynamically at run time depending on their availability, capability, performance, cost, user quality-of-self-service requirement".

"Grid Computing[4] enables virtual organizations to share geographically distributed resources as they pursue common goals, assuming the absence of central location, central control, omniscience, and an existing trust relationship" In other words:

- Grid is a service for sharing computer power and data storage capacity over the Internet and Intranet.

- Grid computing resources has enhanced the performance of computers.

- Computational Grid

  This grid is used to allocate resources specifically for computing power. The resources are usually high-performance machines.

- Data Grid

  This grid is used for housing and providing access to data over multiple organizations.

- Scavenging Grid

  It is one type of Computational grid which uses the unused resources for computational power. It steals CPU cycles when CPU is idle, so it is also called Cycle Scavenging Grid.

1.2 Problem Statement

Grid is service for sharing computer power to execute high performance and high computational Application.The major challenge in such highly heterogeneous and complex computing environment is to design an efficient problem solving environment. The large number of components and their distribution in grid as well as the dynamic structure of the grid increases the risk of failures. Thus providing a scalable and generic problem solving environment as a basic service for grids is of primary importance in such systems.

1.3 Objective

To deal with the faults that are found in grid middleware, customized scheduler algorithm is designed and developed for handling faults so that jobs are scheduled on the basis of available CPU power. Also a new manager is proposed that recovers the jobs from its failure, providing problem solving environment for grid application resolving faults and its impacts.

1.4 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, *Literature survey describe Faults in Grid,Types Of Faults occur in Grid,Survey Of Faults in Grids,Fault Tolerance in Grid Middleware,Faults Tolerance Mechanisms of Grid,Survey Of Faults Tolerance Mechanisms of Grid and Viewpoint*

Chapter 3, *Grid Middleware describe introduction of alchemi grid,Architecture Of Alchemi Based Grid Application,Alchemi Grid Thread Programming Steps,How Alchemi Works,Alchemi Grid Application Life cycle,Alchemi API Class Diagram and Database structure and Thread Execution Flow of alchemi based grid Application.*

Chapter 4, *Grid Set up Configuration describe Experimental Grid Set up Configuration and Alchemi Installation steps and required tools.*

Chapter 5, *Middleware Faults of Alchemi Grid describe Middleware Faults of Alchemi and Present Fault Tolerance Scenario in Alchemi.*

Chapter 6, *Proposed Solution describe Development of Problem Solving Environment,Proposed New Alchemi Manager Concept and Proposed Algorithm.*

Chapter 7, *Implementation describe Alchemi Grid API Main Classes Description,Proposed Algorithm Implementation stpes and Proposed New Alchemi Manager Implementation steps.*

Chapter 8, *Outcome describe Alchemi based Grid Application Result,Proposed New Alchemi Manager Result and Proposed Algorithm Result.*

Chapter 9, *Concluding remarks for the work done. Future goal to use checkpointing mechanism for the solution of identified Middleware Faults of Alchemi Grid.*

Chapter 2

Literature Survey

2.1 Faults in Grid Computing

Grid computing enables the aggregation and sharing of geographically distributed computational data and other resources as a single, unified resource for solving large-scale compute and data intensive applications in dynamic, multiinstitutional virtual organizations. Since Grid resources are highly heterogeneous and dynamic, more faults are likely to occur in Grid environment. The generally accepted definitions of some basic terms as stated in are:
Fault[5]: A fault is a violation of a systems underlying assumptions.
Error[5]: An error is an internal data state that reflects a fault.
Failure[5]: A failure is an externally visible deviation from specifications.
A fault need not result in an error, or an error in a failure. For example, in the Ethernet link layer of the network stack, a packet collision is an error that does not result in a failure because the Ethernet layer handles it transparently.

2.1.1 Types of Faults in Grid

Faults may be classified based on several factors[5].

- Network faults: Faults due to network partition, Packet Loss, Packet corruption.
- Physical faults: Faulty CPUs, Faulty memory, Faulty storage, etc.
- Lifecycle faults: Version faults.

- Media faults: Disk head crashes.
- Processor faults: Machine or operating system crashes.
- Middle ware faults: Middle ware related fault.

2.1.2 Survey of Faults in Grid

In order to identify Survey Of Faults in grids[6], Authors have consulted grid users spread throughout the world through the multiple-choice questions below.

a. What are the more frequent kinds of Faults you face on Grids?

b. What are the mechanisms used for detecting and/or correcting and/or tolerating faults?

The questionnaire was sent several grid User in all over the world, such as : The main kinds of Faults (see figure 2.1) are related to the environment configuration. Almost 76% of the responses have pointed this out. According to some people surveyed, the lack of control over grid resources is the main source of configuration failures. Following this, we have middleware failures with 48%, application failures with 43% and finally hardware failures with 34%.

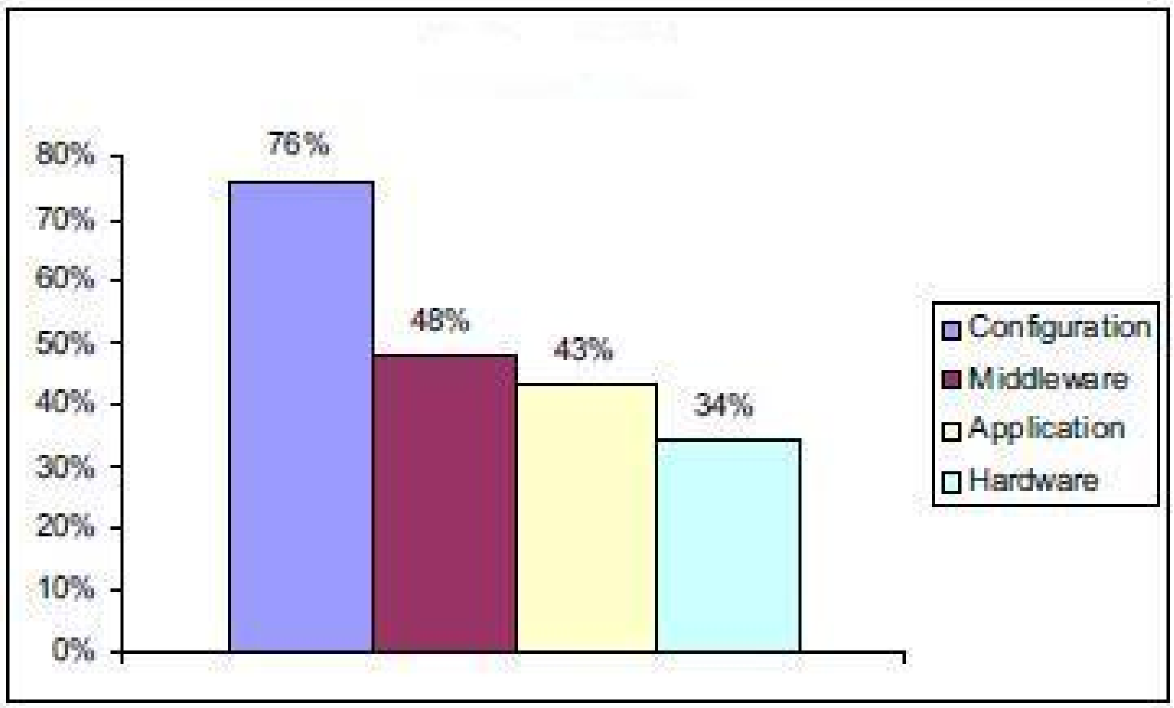


Figure 2.1: Kinds of Faults in Grid[6]

2.2 Fault Tolerant Technique in Grid

In a grid environment there are potentially thousands of resources, services and applications that need to interact in order to make possible the use of the grid as an execution platform.Since these elements are extremely heterogeneous, there are many failure possibilities, including not only independent failures of each element, but also those resulting from interactions between them. Because of the inherent instability of grid environments, fault detection and recovery is a critical component that must be addressed. The need for fault tolerance is especially acute for large parallel applications since the failure rate grows with the number of processors and the duration of the computation. The earliest use of computers made it apparent that even with careful design and good components, physical defects and design errors were unavoidable. Thus, designers of early computers used practical techniques to increase reliability. They used redundant structures to mask failed components; error control codes and duplication or triplication with voting to detect or correct information errors; diagnostic techniques to locate failed components and automatic switchovers to replace failed subsystems.

2.2.1 Fault Tolerance in Grid

Fault tolerance[5] is the survival attribute of computer systems. The function of fault tolerance is

"*to preserve the delivery of expected services despite the presence of fault-caused errors within the system itself. Errors are detected and corrected, and permanent faults are located and removed while the system continues to deliver acceptable service*"

Fault tolerance is the property of a system that continues operating consistent with its specifications even in the event of failure of some of its parts. From a user's point of view, a distributed application should continue despite failures. The fault tolerance has become the main topic of research. Till now there is no single system that can be called as the complete system that will handle all the faults in grids. Grid is a dynamic system and the nodes can join and leave voluntarily. For making fault tolerance system a success, following point must consider:

- How new nodes join the system,
- How computing resources are shared,
- How the resources are managed and distributed

2.2.2 Fault Tolerance in Grid Middleware

Grids have middleware stacks, which are a series of cooperating programs, protocols and agents designed to help users access the resources of a Grid. Grid Middleware refers to the security, resource management, data access, instrumentation, policy, accounting, and other services required for applications, users, and resource providers to operate effectively in a Grid environment. Middleware acts as a sort of 'glue', which binds these services together. Middleware connect applications with resources. Formally Grid middleware can be defining as: "*A mediator layer that provide a consistent and homogeneous access to resources managed locally with different syntax and access methods*" The brief overview of the some popular middleware Alchemi, Globus, Sun Engine and their fault tolerance mechanisms has been discussed in this section.

Alchemi:Alchemi[9] is a .NET-based grid computing framework that provides the run- time machinery and programming environment required to construct desktop grids and develop grid applications. Till today not much work is done on Alchemi Fault tolerance. The basic technique used by Alchemi for Fault Tolerance is Heartbeat Mechanism.The Executors in the Alchemi sends the heartbeat messages to the Manager at regular interval of time. Manager after receiving messages from executors guesses that the executor node is still working. So the procedure of fault tolerance is not so well planned. In later chapters we will see fault tolerance procedure of Alchemi in more detail and try to find challenges in its fault tolerance system.

Sun Grid Engine:Sun's Grid Engine[20] is a Distributed Resource Management tool; it provides load management across heterogeneous, distributed computing environments. Sun made the source available under the Grid Engine project . The Grid Engine project is an open source version of the software is known as Grid Engine. Grid Engine is generally installed across a cluster of machines using a shared file system. It can be configured to work across disparate file systems. Users submit jobs to Grid Engine and Grid Engine manages the allocation of jobs to machines within the cluster. This ensures the resources are used more productively therefore increasing availability. In this middleware with the help of the user and kernel level Check pointing manage the fault.

Globus:The Globus toolkit[19] is designed to enable people to create computational Grids. Globus is an open source initiative aimed at creating new Grids capable of the scale of computing seen only in supercomputers up to now. As an open source project any person can download the software, examine it, install it and hopefully improve it. By this constant stream of comments and improvements, new versions of the software can be developed with increased functionality and reliability. In this way the Globus project itself will be on going with constant evolution of the toolkit.Globus Toolkit has three pyramids of support built on top of a security infrastructure, as illustrated. They are:Resource management,Data management,Information services,All of these are built on top of the underlying Grid Security Infrastructure (GSI).This provides security functions, including single/mutual authentication, fault tolerant, confidential communication, authorization, and delegation. Through the monitoring of the system by the gate keeper we can manage the fault. It provide secure communication between clients and servers. It also communicates with the GRAM client (globus run) and authenticates the right to submit jobs. After authentication, gatekeeper forks and creates a job Manager delegating the authority to communicate with clients.

2.2.3 Fault Tolerance Mechanisms of Grid

grid users have used automatic ways to deal with failures in their Grid Environment. To achieve the automatic ways to deal with failures, various fault tolerance mechanisms are there. Some of these fault tolerance mechanisms[5] are:

Application-dependent: Grids are increasingly used for applications requiring high levels of performance and reliability, the ability to tolerate failures while effectively exploiting the resources in scalable and transparent manner must be integral part of grid computing resource management systems.

Monitoring Systems: In this a fault-monitoring unit is attached with the grid. The base technique which most of the monitoring units follow is heart beating technique.

Fault Tolerant Scheduling: With the momentum gaining for the grid computing systems, the issue of deploying support for integrated scheduling and fault-tolerant approaches becomes a paramount importance. For this most of the fault tolerant scheduling algorithms are using the coupling of scheduling policies with the job replication schemes such that jobs are efficiently and reliably executed. Scheduling policies are further classified on basis of time-sharing and space sharing.

Checkpointing-recovery: Check pointing and rollback recovery provides an effective technique for tolerating transient resource failures, and for avoiding total loss of results. Check pointing involves saving enough state information of an executing program on a stable storage so that, if required, the program can be re-executed starting from the state recorded in the checkpoints. Checkpointing distributed applications is more complicated than Checkpointing the ones, which are not distributed. When an application distributed, the Checkpointing algorithm not only has to capture the state of all individual processes, but it also has to capture the state of all the communication channels effectively.

2.2.4 Survey of Fault Tolerance Mechanism of Grid

Grid users have used automatic ways to deal with failures on their systems (see figure 2.2). Nevertheless, 57% of them are application dependent.Even when monitoring systems are used (29% of the cases).Checkpointing is used in 29% of the systems and fault-tolerant scheduling in 19%. In some cases, different mechanisms are combined.In case of checkpointing-recovery and fault-tolerant scheduling, they are only able to deal with crash failure semantics for both hardware and software components.

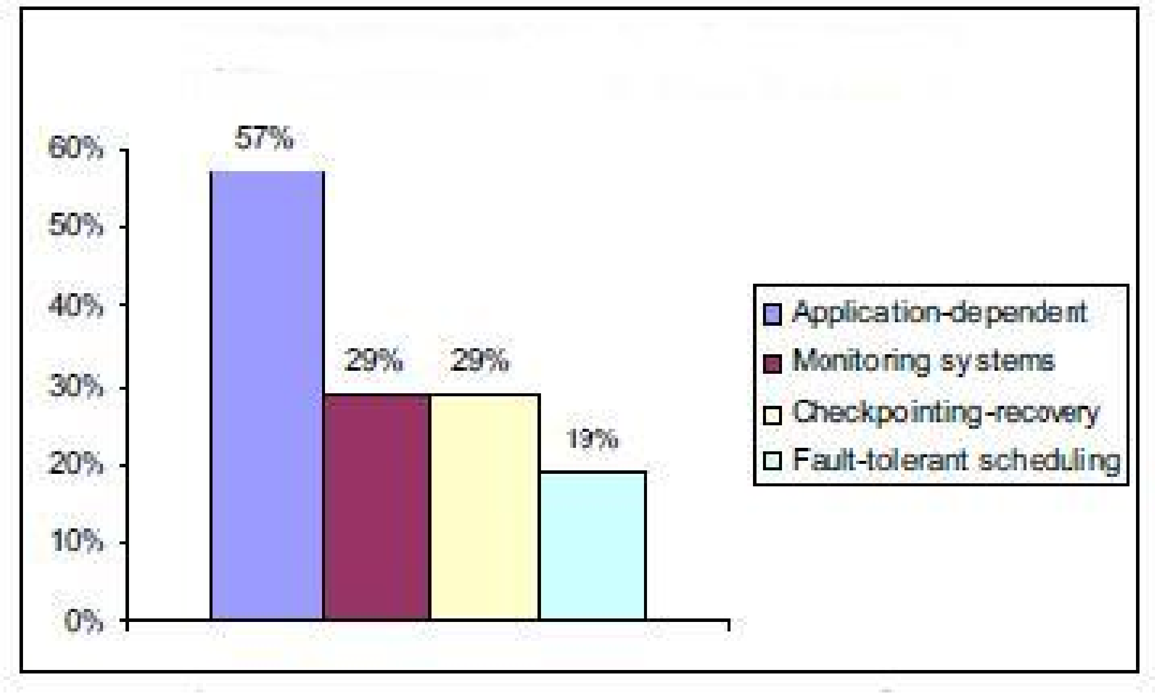


Figure 2.2: Fault Tolerance Mechanism in Currant Use[6]

2.3 Viewpoint

After analyzing the need of fault handling in grid,realized to solve middleware related faults.Studied different middleware Alchemi,Globus,Sun Grid Engine etc.For verification of system,grid based application was developed in alchemi middleware.Based on various result identified alchemi middleware related faults are realized.Alchemi supports scheduling algorithm which selects highest priority thread among the pool of threads and dedicate to it to the available executor keeping scope for enhancement of application performance. This thesis proposes customized scheduler algorithm for handling faults so that scheduling of jobs will be based on available CPU power.Also proposed new manager recovers the jobs from its failure providing, problem solving environment for grid application resolving faults and its impacts and integrated with Alchemi grid.

Chapter 3

Grid middleware

3.1 Why Alchemi middleware?

Alchemi[9][16][12] is an open source software framework that allows you to painlessly aggregate the computing power of networked machines into a virtual supercomputer And develop applications to run on the grid. Alchemi includes: The runtime machinery (Windows executables) to construct grids. A .NET API and tools to develop .NET grid applications and grid enabled legacy applications. It has been designed with the primary goal of being easy to use without sacrificing power and flexibility. As Alchemi is the emerging technology, so a lot of work has to be done to make it a standard .NET based middleware. We are using Alchemi as the framework to setup the grids. There can be many reasons for choosing Alchemi as the framework for building the computational grid and then further choosing to build a fault tolerant system for it. Some of these reasons are:

- Alchemi is the first .NET based stable grid.
- Most important is that Alchemi is open source. So one can do any number of changes in it.
- Another important reason is that most of the systems in our labs are running Windows Operating Systems.
- Fault tolerance research area is still under development in Alchemi.

3.2 Architecture of Alchemi Based Grid Application

Alchemi grid architecture[8] is complex comprising of many layers as shown in figure 3.1. Typically,a user application instantiates 3 classes of objects from theAlchemi API: GConnection, GApplication and one or more GThreads. GConnection is a class responsible for connecting and sending authentication information from the user application to the central component called Manager.
The GConnection is consumed by GApplication, which is also created by the user application. To take an advantage of the grid, user application must create a number of independent jobs and submit them to the grid.

In Alchemi terminology, the independent jobs are referred to as grid threads. They are objects whose class definition is inherited from GThread class. Before executing GApplication Start() method, which starts parallel execution of the grid threads, the GThreads are added to the GApplication thread collection.

Furthermore, a DLL or EXE module that implements this class, along with all dependant modules are added to the manifest of the GAplication instance. Since the threads need to be sent to participating grid nodes, a thread class is marked with Serializable attribute, which is a .Net concept for marking classes available for serialization.

A serialization is a process of transforming an object into a form which can be saved into a file or other storage or be sent across a wire. The GApplication class defines a number of events, the most important being ApplicationFinish. The Application Finish event is raised when all dispatched threads reach the finished state.

In Alchemi terminology, the central controller is called Manager. The Manager is responsible for persisting GThread objects received from the user application. The other roles of the manager include GThread scheduling, dispatching and monitoring.
Grid nodes capable of code execution are called Executors. The executors register themselves with the manager. They can be dedicated or non dedicated participants of the

Alchemi grid. The manager monitors a state and availability of the registered executors. The executor states are persisted into short term storage. Scheduler is a component responsible for assigning the GThreads to the executors. In the Alchemi environment it is possible to plug-in a custom scheduler.

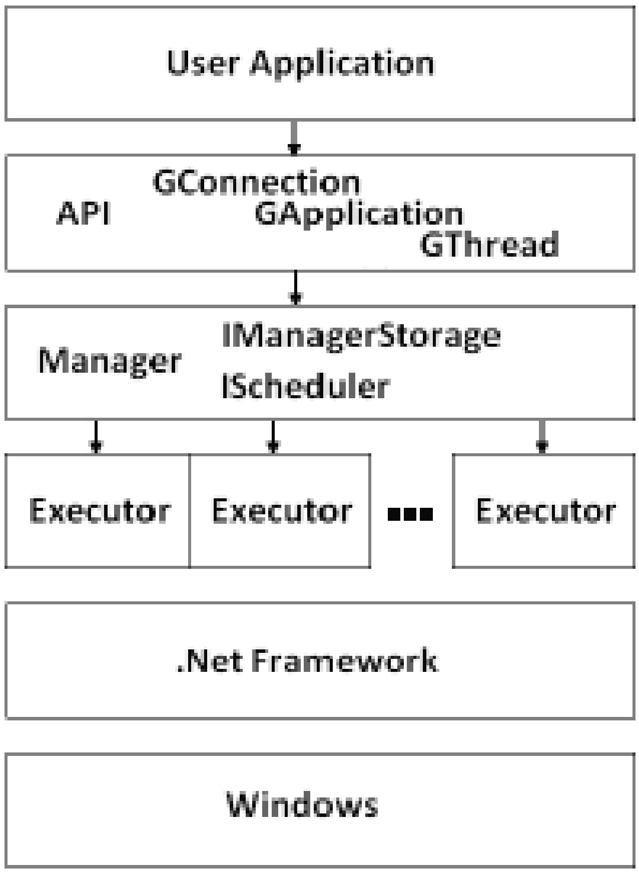


Figure 3.1: Alchemi Grid Architecture[8]

3.3 Alchemi Grid Thread Programming Steps

The two central classes in Alchemi grid API are GThread and GApplication that represent grid thread and application respectively.Gthread class is used for the code to be executed remotely and GApplicaion is for locally executed code. To build the custom grid thread,following steps[16] are to be followed:

- import the namespace Alchemi.Core
- Extend the GThread class
- Override the start() method
- Make the class serializable

To build an appplicaion,following steps are to be followed:

- Import the namespace Alchemi.Core
- Create an instance of GApplication with manager's IP and port no
- Add custom grid thread module as a dependency
- Create an instances of grid thread
- Add the thread instances to the GApplication's Threads Collection
- Set the finish callback method of grid
- Set the finish callback method of applicaion
- Start the application
- Write the callback methods for thread and application

3.4 How Alchemi Works

There are three types of distributed components (nodes) involved in the construction of Alchemi grid (shown in figure 3.2) and execution of grid applications[9][16]: Manager, Executor and User.

a. Manager: The Manager manages the execution of grid application and provides services associated with managing thread execution.The Executors register themselves with the Manager which in turn keeps track of their availability. Threads received from the owner are placed in a pool and scheduled to be executed on the various available Executors.A priority for each thread can be explicitly specified when it is

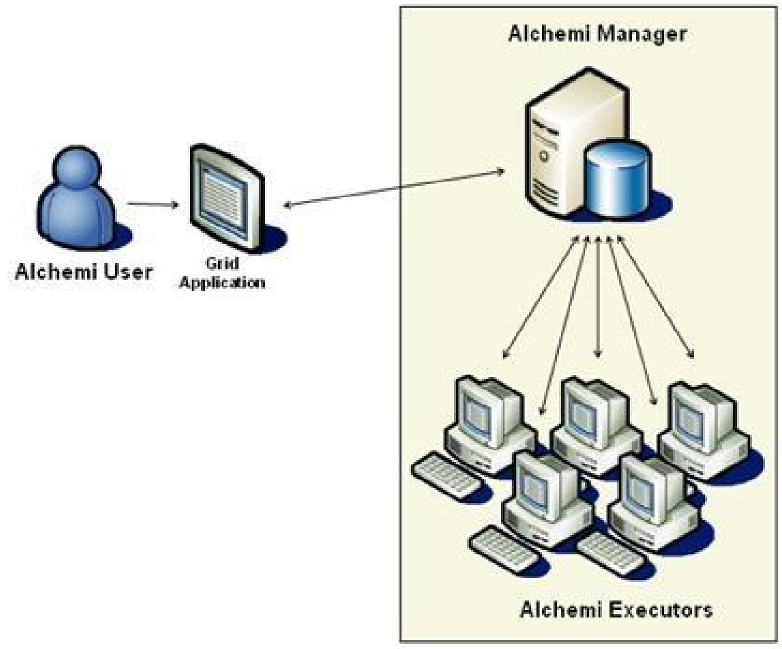


Figure 3.2: An Alchemi Grid[9]

created within the Owner,but is assigned the highest priority by default if none is specified.Threads are scheduled on a Priority and First Come First Served(FCFS) basis,in that order. The Executors return completed threads to the Manager which are subsequently passed on or collected by the respective Owner.

b. Executor: The Executor accepts threads from the Manager and executes them. An Executor can be configured to be dedicated, meaning the resource is centrally managed by the Manager, or non-dedicated, meaning that the resource is managed on a volunteer basis via a screen saver or explicitly by the user. For non-dedicated execution, there is one-way communication between the Executor and the Manager. In this case, the resource that the Executor resides on is managed on a volunteer basis since it requests threads to execute from the Manager. When two-way communication is possible and dedicated execution is desired the Executor exposes an interface so that the Manager may communicate with it directly. In this case, the Manager explicitly instructs the Executor to execute threads, resulting in centralized management of the resource where the Executor resides.

c. User : Grid applications are executed on the User node. The API abstracts the implementation of the grid from the user and is responsible for performing a variety of services on the users behalf such as submitting an application and its constituent threads for execution, notifying the user of finished threads and providing results and notifying the user of failed threads along with error details.

3.5 Alchemi Grid Application Life cycle

To develop and execute a grid application[9][16] the developer creates a custom grid thread class that derives from the abstract GThread class. An instance of the GApplication object is created and any dependencies required by the application are added to its Dependency Collection. Instances of the GThread derived class are then added to the GApplications ThreadCollection. The lifecycle of a grid application is shown in Figure, showing simplified interactions between the Owner and Executor nodes respectively and the Manager. The GApplication serializes and sends relevant data to the Manager, where it is persisted to disk and threads scheduled. Application and thread state is maintained in a SQL Server / MSDE database.Non-dedicated executors poll for threads to execute until one is available. Dedicated executors are directly provided a thread to execute by the Manager. Threads are executed in .NET application domains, with one application domain for each grid application. If an application domain does not exist that corresponds to the grid application that the thread belongs to, one is created by requesting, destabilizing and dynamically loading the applications dependencies. The thread object itself is then desterilized, started within the application domain and returned to the Manager on completion. After sending threads to the Manager for execution, the GApplication polls the Manager for finished threads. A user-defined GThreadFinish delegate is called to signify each threads completion and once all threads have finished a user-defined GApplicationFinish delegate is called.

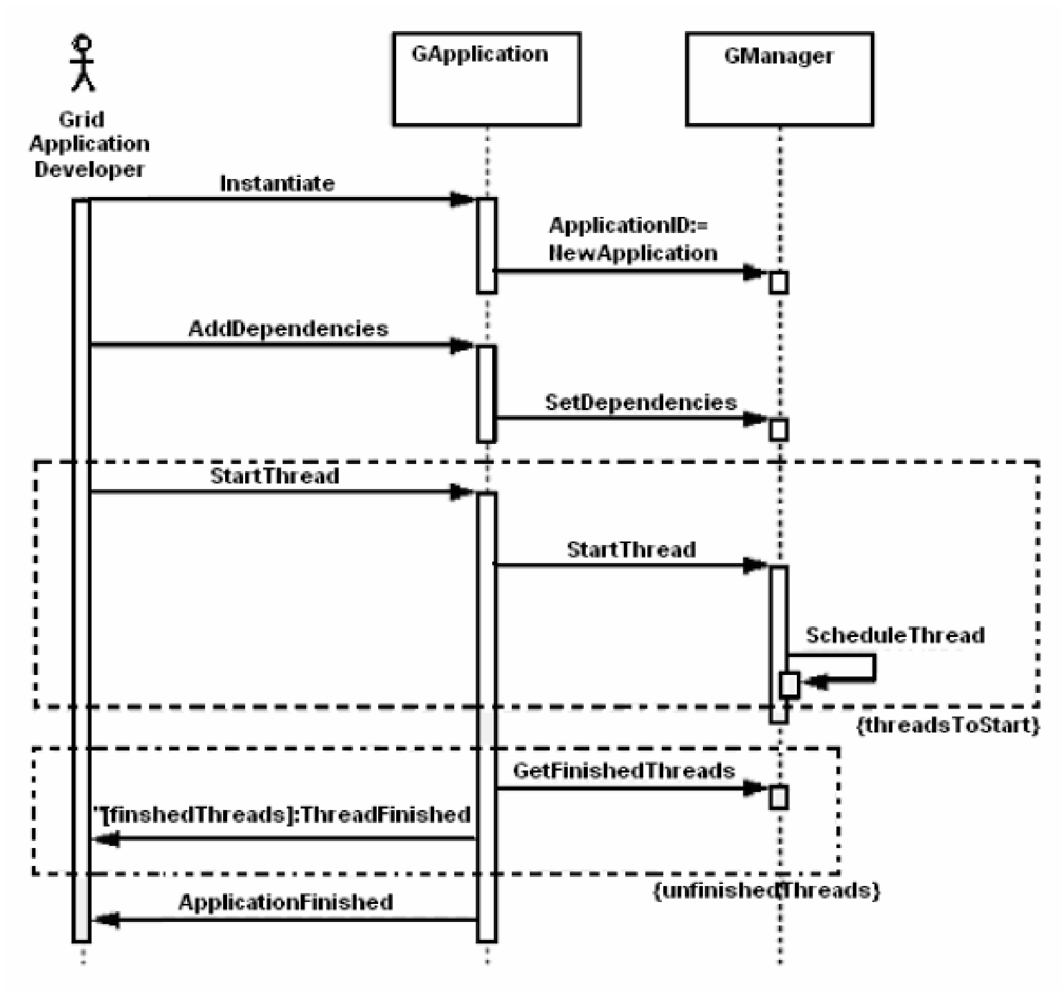


Figure 3.3: Interaction between User and Manager nodes[9]

3.6 Alchemi API Class Diagram and Database structure

this section describes the database architecture[9][16][11] of alchemi in which relevant information is stored. Some tables are static storing data structures are grp,prm,usr and prm which stores information about the users, groups and permissions. There are also some dynamic data structures which are reflected to update the recent information. They are: application,executor,thread.also describes Class Diagram of Alchemi API[9][16] and main Classes[9][16] Description of Alchemi.

column name	**grid application table description**
applicationid	Unique identifier for application generated by manager.
timecreated	Time when the application was created.
usrname	Identifies the user by whom the application is supplied.
applicationname	Name of the application.
timecompleted	Completed time of application.

Table I: Alchemi grid application table description

column name	**executor table description**
executorid	Unique identifier for executor generated by manager.
isdedicated	Mode of an executor Dedicated :1 ,Non-dedicated :0.
isconnected	Whether the executor is connected or not.
pingtime	Time of last ping to the executor.
host	Host name.
port	Port.
usrname	Executor User Name.
cpumax	max cpu.
cpuusage	usage of cpu.
cpuavail	Availableof total of cpu.
cputotal usage	Total cpu usage.
memmax	Total Memory size.
diskmax	Total Disk size.
numcpus	No of cpu.
cpuLimit	Limit to usage of cpu.
memLimit	Limit to the Memory.
diskLimit	Limit on disk Usage.
arch	Architecture.
os	Operating System.

Table II: Alchemi executor table description

column name	**thread table description**
internalthreadid	Thread ID assigned by system .
applicationid	application id to which the thread relates.
executorid	executor on which the thread is executed.
threadid	thread ID assigned by user unique for application.
state	current state of the thread.
timestarted	time on which thread execution start.
timefinished	time on which thread execution finished.
priority	priority of the thread .
failed	whether the execution of thread is failed .

Table III: Alchemi thread table description

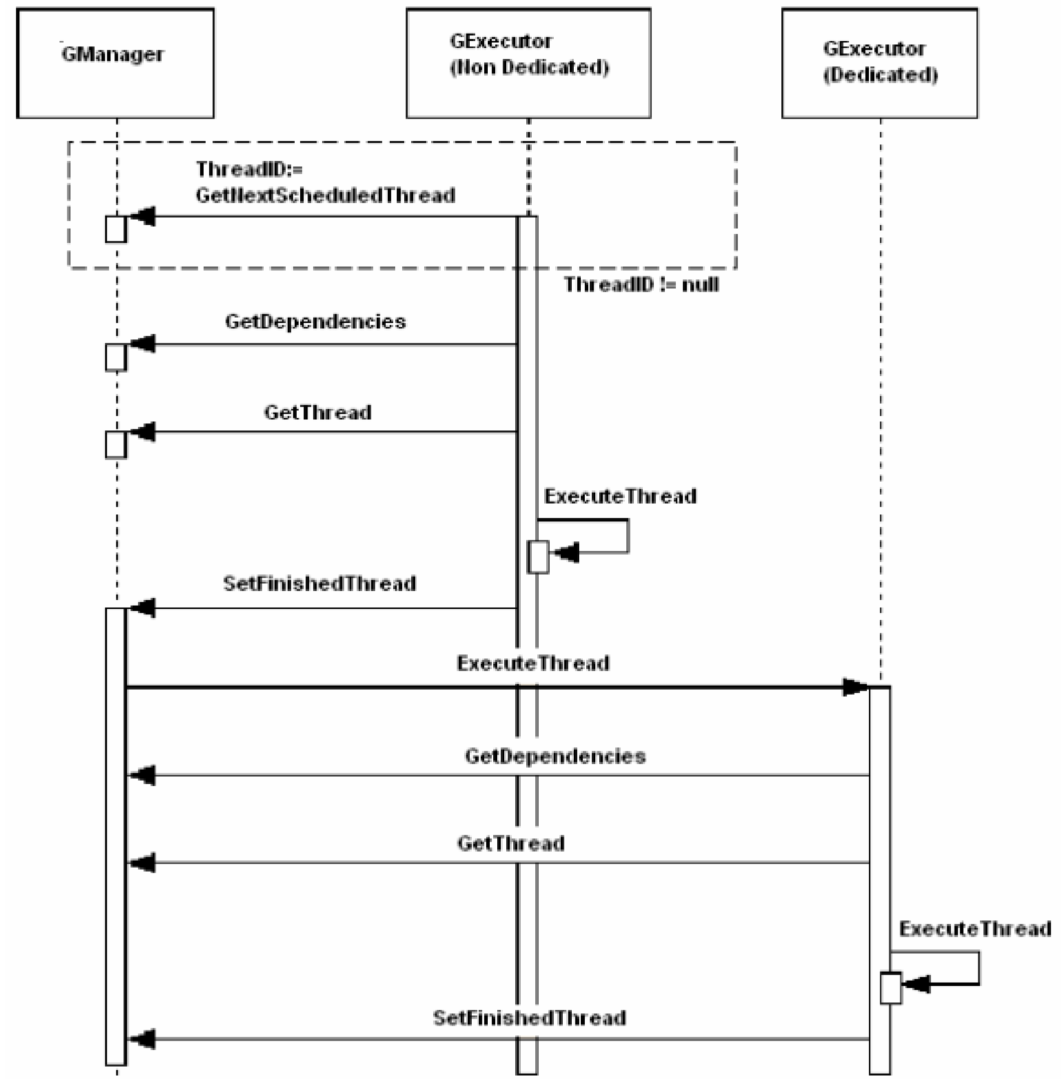


Figure 3.4: Interaction between Executor and Manager nodes[9]

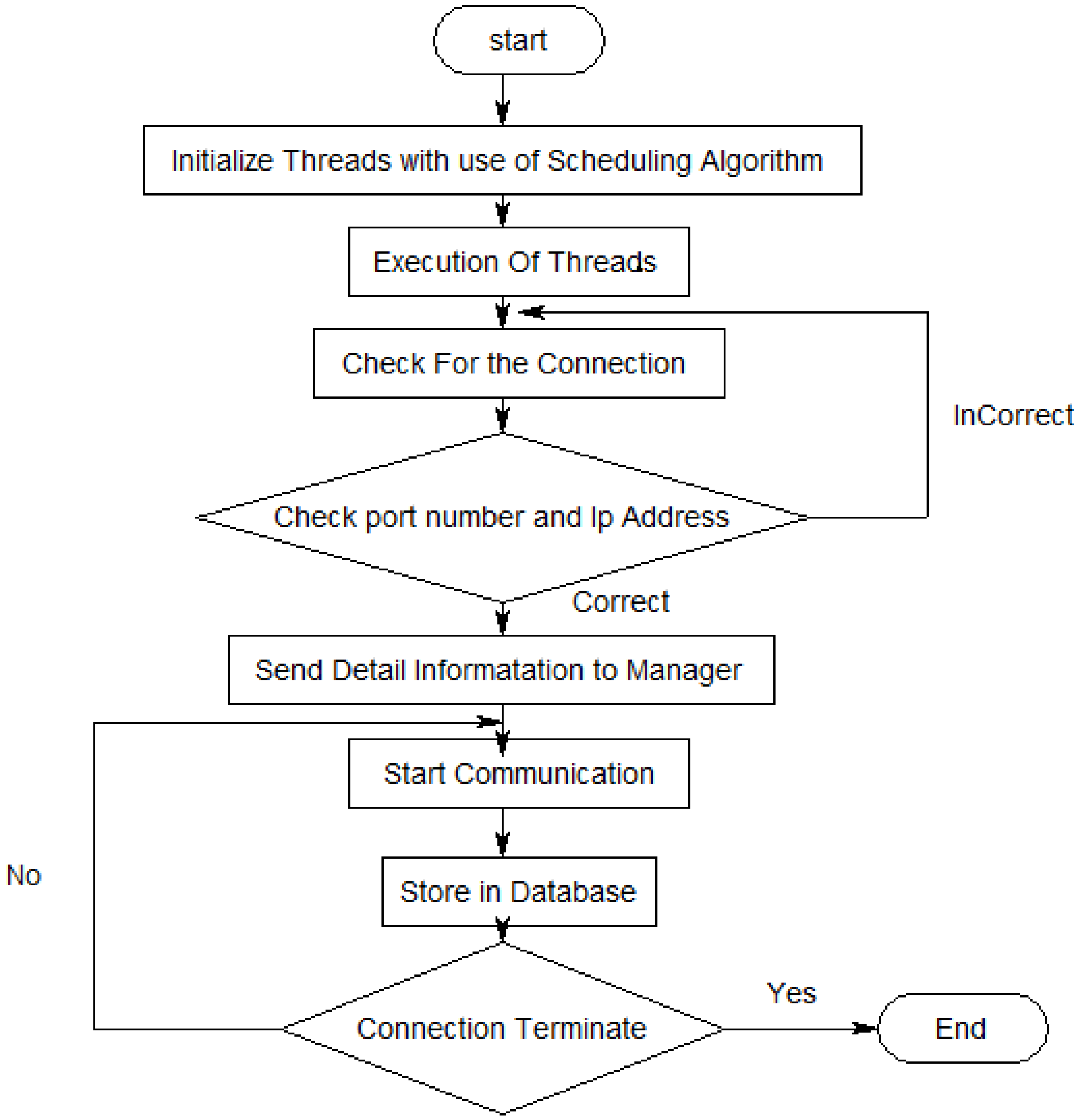


Figure 3.5: Thread Execution Flow of alchemi based grid Application[9]

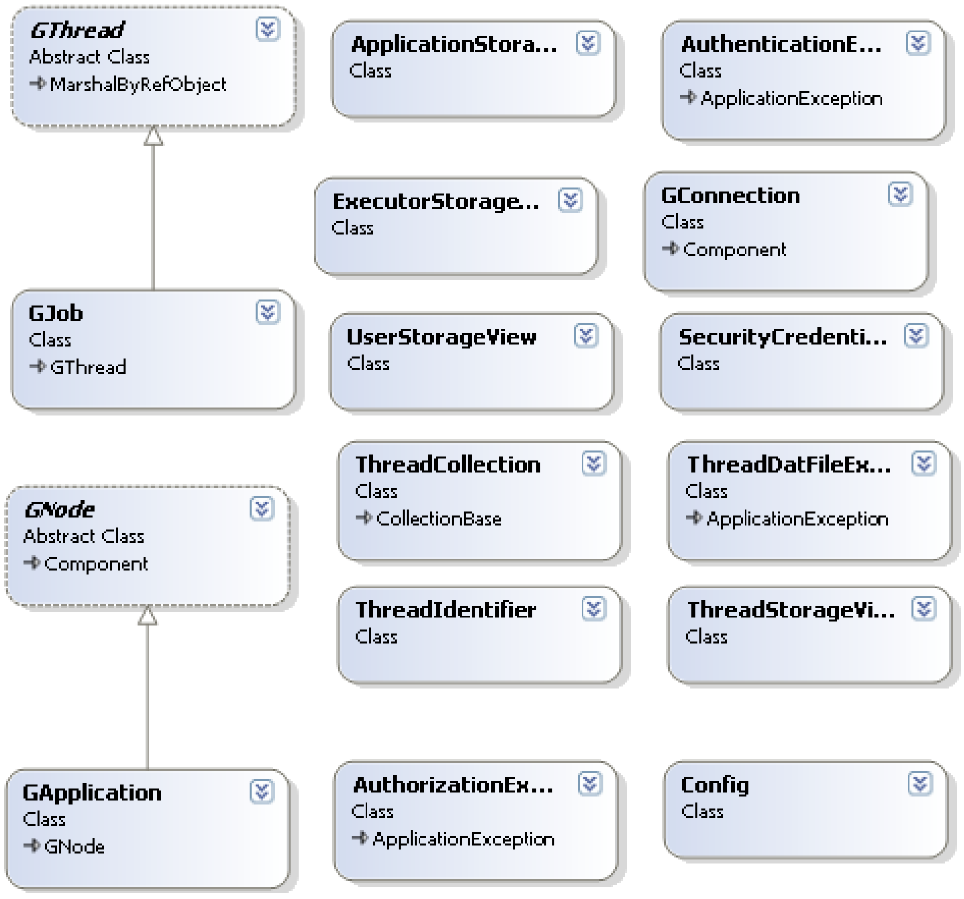


Figure 3.6: Alchemi Core class Diagram

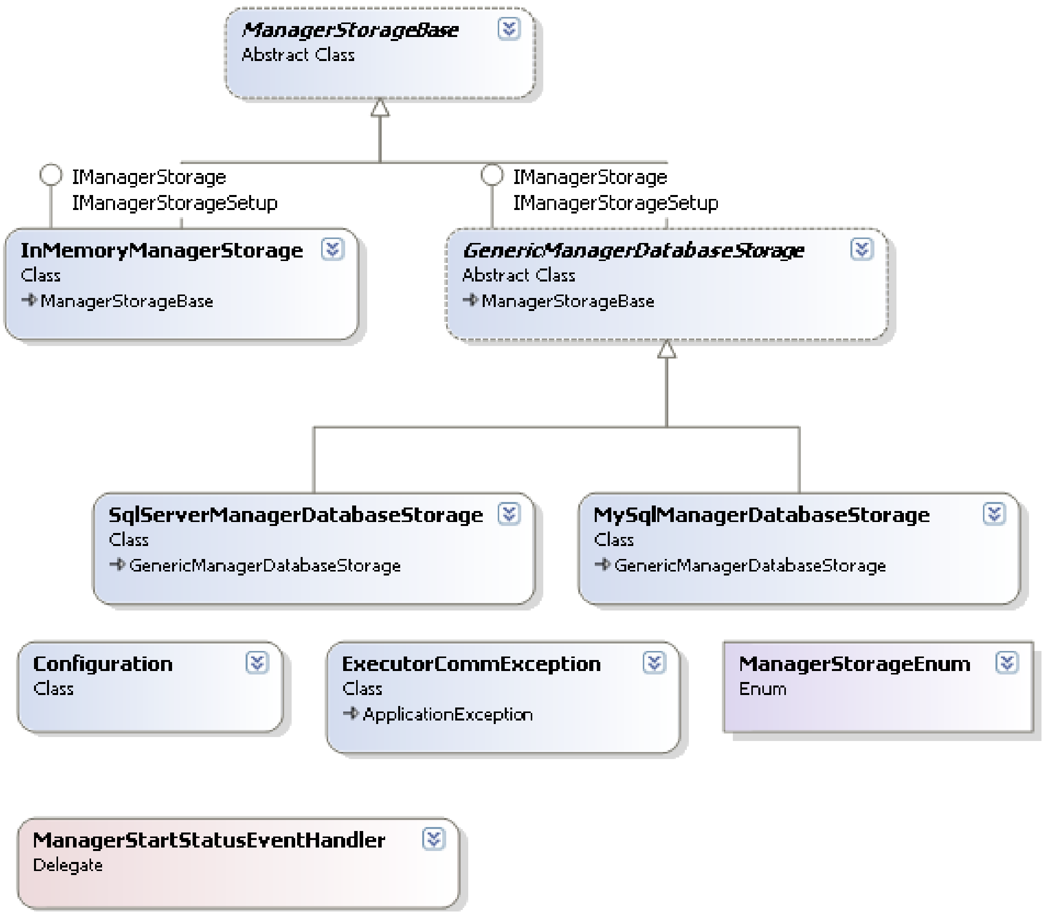


Figure 3.7: Alchemi Manager Class Diagram 1

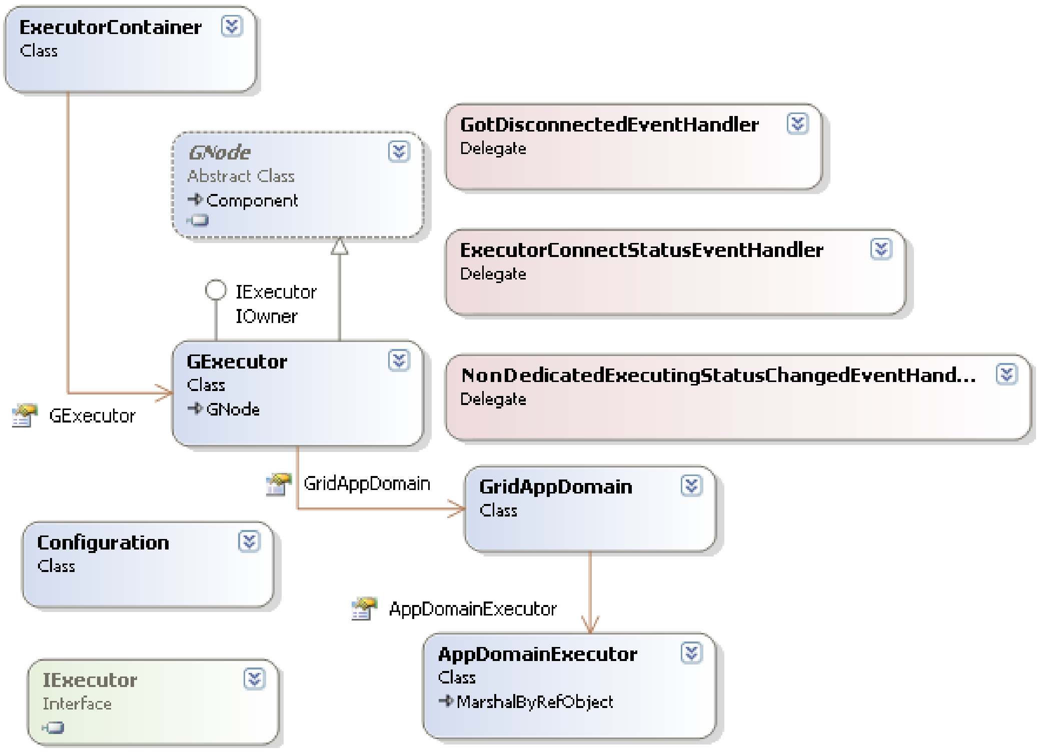


Figure 3.8: Alchemi Executor Class Diagram

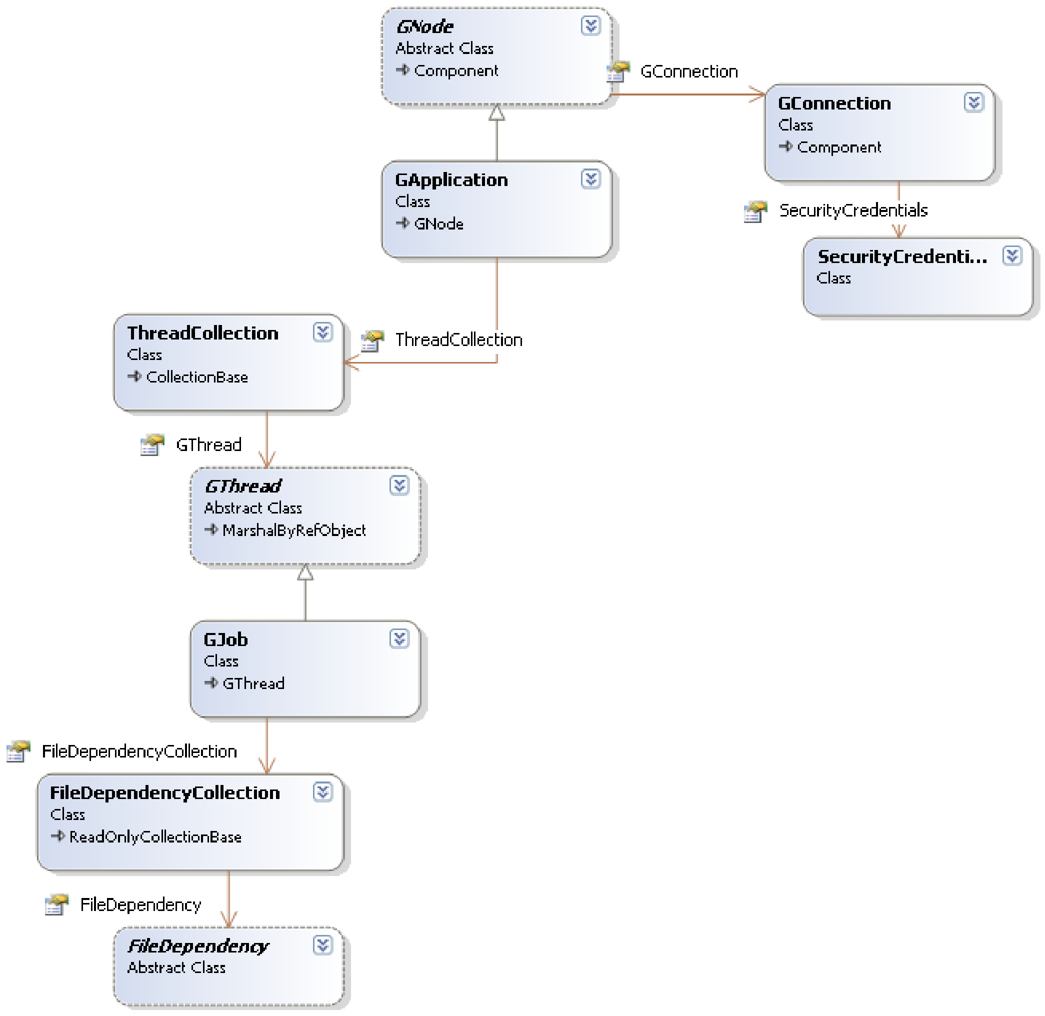


Figure 3.9: Alchemi Grid Application Class Diagram

Chapter 4

Grid Set up Configuration

4.1 Experimental Grid Set up Configuration

Because of lack of availability of grid simulators, Grid setup[15] is required to configure. This grid is scalable and no of nodes can be connected in grid. However to deal with our algorithm, 2 machines are configured in grid.

4.1.1 Alchemi Installation

Alchemi Manager

The prerequisite for Alchemi Manager[15] Installation are:

Front End Tools

- Microfost Visual Studio 2005
- Alchemi 1.0.6

Machine Name	Ip Address	CPU Clock rate	RAM
Pgcse-14	10.1.3.14	3.20 GHZ	1 GB
Pgcse-15	10.1.3.15	3.20 GHZ	1 GB
Pgcse-16	10.1.3.16	2.20 GHZ	1 GB
Pgcse-17	10.1.3.17	2.20 GHZ	1 GB
Pgcse-18	10.1.3.18	3.20 GHZ	1 GB

Table I: Experimental Grid Set up Configuration

Back End Tools

- Microsoft Sql Server 2005

Alchemi Executer

The only prerequisite for Alchemi Executer[15] Installation are:

- .Net Framework 2.0

Chapter 5

Middleware Faults of Alchemi Grid

5.1 Middleware Faults of Alchemi

1. If the manager node fails [10][17] due to disconnected from the network or hardware failure then all the executor nodes will probably stop their ongoing work. Again if the manager node and the database resides on the same computer then as soon as the manager node fails there is no way to retrieve the output of the already processed sub tasks. Hence the failure of the Grid manager may seriously hamper the performance of a Grid environment so grid application execution fails. (shown in figure-5.1)

2. If any of the executor nodes fails[10][17][13] to complete its assigned job then this task is rescheduled to another executor and the task is executed again from its initial state. Generally, if any job associated with a Grid application unable to succeed during its runtime in the executor then the scheduling algorithm just marks the job as failed and reset but it does not keep any track of the percentage of the work that has already been done.so if some Executer has failed after 1 hours of work then all the work done during that 1 hours will be wasted.

3. Manager node uses FIFO policy for scheduling. It stores the threads according to their priority It does not consider the CPU load of the processors on which the executors are running. If more than one executor is available at a time, it might happen that a thread is scheduled on a more loaded executor[17] which can degrade the performance.

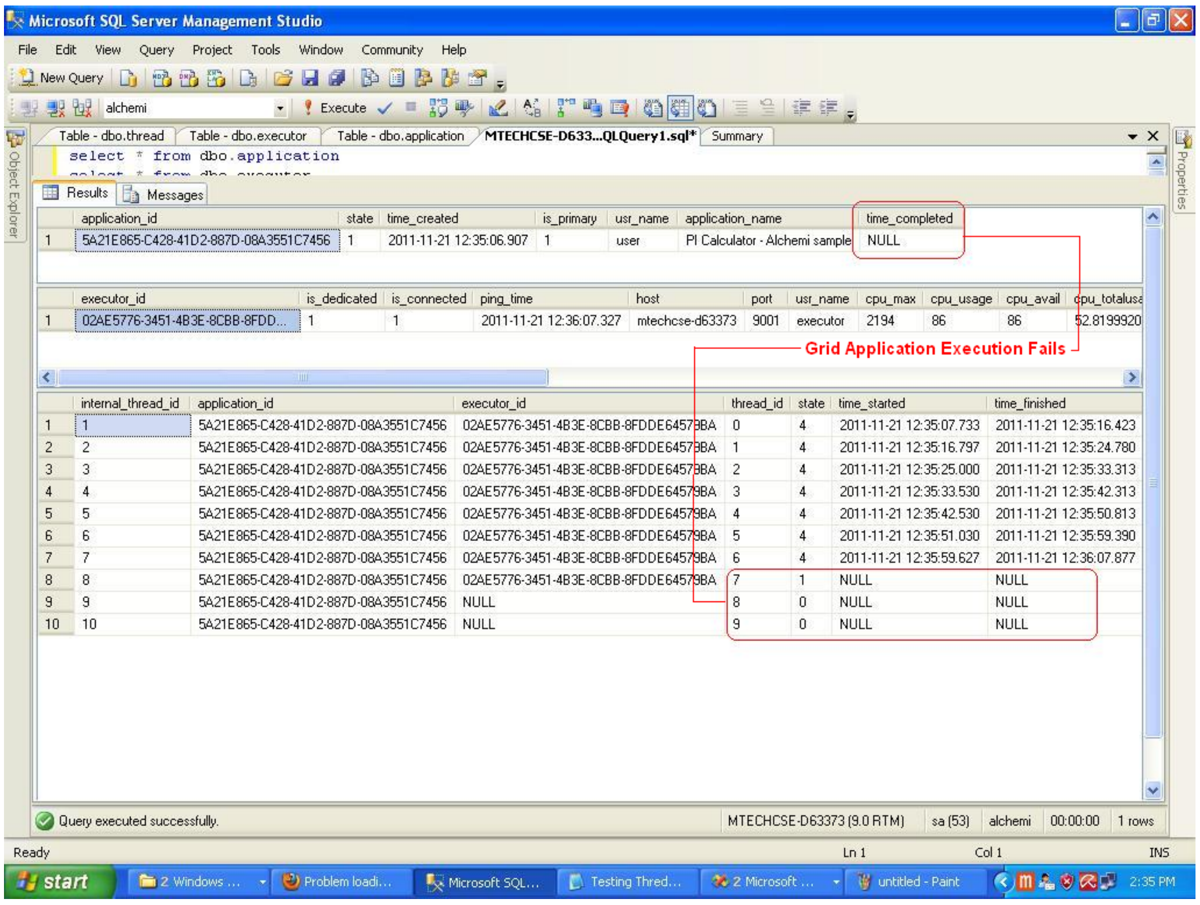
5.2 Present Fault Tolerance Scenario in Alchemi

1. Heartbeat mechanism[7] is used by Alchemi: Executor nodes send the heartbeat messages at some interval to the Manager to whom they are connected. This will help the Manager node to maintain the status of the Executor nodes. In case the Manager node doesn't receive the heartbeat messages from the executor nodes between the pre-decided times, the Manager node consider that executor node to be dead and update its information.(shown in figure-5.2)

2. Executor fails due to a hardware failure.In this case Alchemi is using the heartbeat technique to re- schedule the thread to another Executor.

3. Executor fails due to the user logging off or stopping down the Executor. In this case the Manager is informed that the Executor is going offline and the thread is re-scheduled.

4. Dynamically determine new nodes added or deleted: As the new node is added or removed, the Manager is being provided with the information dynamically and it updates the console information to make a correct record of the information so as to avoid any faults.(shown in figure-5.3)



```
select * from dbo.application
```

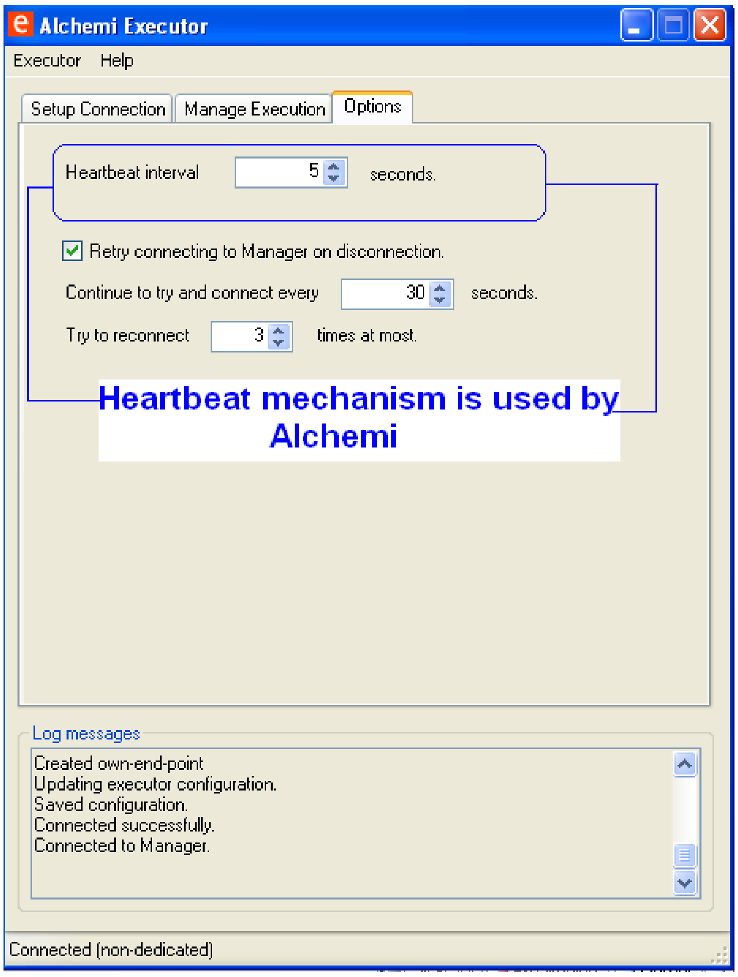
	application_id	state	time_created	is_primary	usr_name	application_name	time_completed
1	5A21E865-C428-41D2-887D-08A3551C7456	1	2011-11-21 12:35:06.907	1	user	PI Calculator - Alchemi sample	NULL

	executor_id	is_dedicated	is_connected	ping_time	host	port	usr_name	cpu_max	cpu_usage	cpu_avail	cpu_totalusa
1	02AE5776-3451-4B3E-8CBB-8FDD...	1	1	2011-11-21 12:36:07.327	mtechcse-d63373	9001	executor	2194	86	86	52.8199920

	internal_thread_id	application_id	executor_id	thread_id	state	time_started	time_finished
1	1	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	0	4	2011-11-21 12:35:07.733	2011-11-21 12:35:16.423
2	2	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	1	4	2011-11-21 12:35:16.797	2011-11-21 12:35:24.780
3	3	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	2	4	2011-11-21 12:35:25.000	2011-11-21 12:35:33.313
4	4	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	3	4	2011-11-21 12:35:33.530	2011-11-21 12:35:42.313
5	5	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	4	4	2011-11-21 12:35:42.530	2011-11-21 12:35:50.813
6	6	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	5	4	2011-11-21 12:35:51.030	2011-11-21 12:35:59.390
7	7	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	6	4	2011-11-21 12:35:59.627	2011-11-21 12:36:07.877
8	8	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	7	1	NULL	NULL
9	9	5A21E865-C428-41D2-887D-08A3551C7456	NULL	8	0	NULL	NULL
10	10	5A21E865-C428-41D2-887D-08A3551C7456	NULL	9	0	NULL	NULL

```
Query executed successfully.    MTECHCSE-D63373 (9.0 RTM)    sa (53)    alchemi    00:00:00    1 rows
```

Figure 5.1: Failure of Grid Application



Heartbeat interval 5 seconds.

[✓] Retry connecting to Manager on disconnection.

Continue to try and connect every 30 seconds.

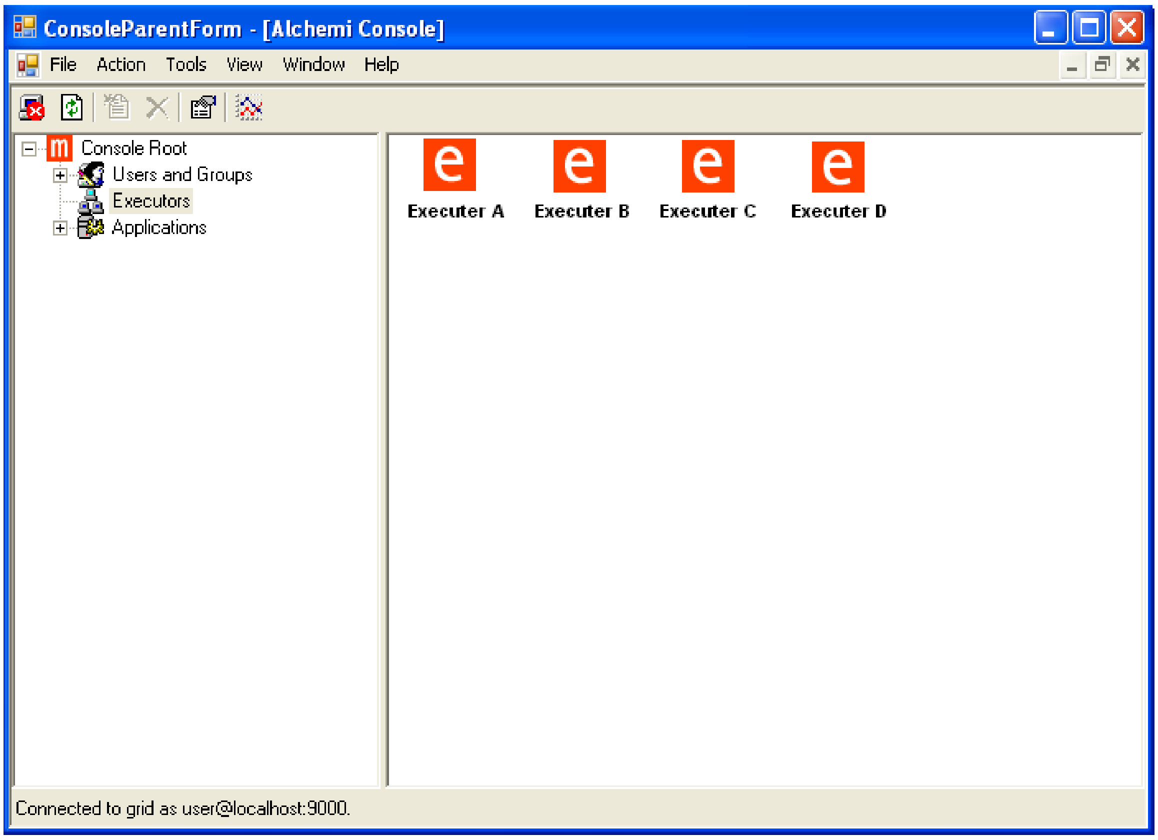
Try to reconnect 3 times at most.

Heartbeat mechanism is used by Alchemi

```
Created own-end-point
Updating executor configuration.
Saved configuration.
Connected successfully.
Connected to Manager.
```

Connected (non-dedicated)

Figure 5.2: Heartbeat mechanism of Alchemi[16]



Connected to grid as user@localhost:9000.

Figure 5.3: Console information in Alchemi[16]

Chapter 6

Proposed Solution

6.1 Development of problem solving environment in grid

Identified alchemi middleware faults has been solved using “Problem Solving Environment” shown in figure 6.3.If the manager node fails [10][17] due to disconnected from the network or hardware failure then all the executor nodes will probably stop their ongoing work. Again if the manager node and the database resides on the same computer then as soon as the manager node fails there is no way to retrieve the output of the already processed sub tasks. Hence the failure of the Grid manager may seriously hamper the performance of a Grid environment so grid application execution fails. so we are proposed “New Alchemi Manager” shown in figure 6.1.If more than one executor is available at a time, it might happen that a thread is scheduled on a more loaded executor so threads execution time increase which can degrade the performance of Alchemi grid so we are proposed “Customized Scheduler Algorithm” shown in figure 6.2.Proposed “New Alchemi Manager” and “Customized Scheduler Algorithm” embedded in problem solving environment and integrated with alchemi grid to handle faults leading towards performance enhancement.

6.1.1 Proposed New Alchemi Manager

Alchemi used “Heartbeat mechanism” :Executors send the heartbeat messages at some interval(5 second) to the Manager to whom they are connected. This will help the Manager to maintain the status of the Executors. In case the Manager doesn’t receive the heartbeat messages from the executor between the pre-decided times, the Manager consider that executor to be dead and update its information, with use of "Heartbeat mechanism" we proposed "New Alchemi Manager". After every heartbeat interval, the Alchemi Manager sends a heartbeat messages to the New Alchemi manager. The New Alchemi manager is on the one of the registered executor. In case of absence of heartbeat messages for certain predefined time interval lets the New Alchemi manager to consider the Alchemi Manager has failed, and it itself takes the control as new Alchemi manager.

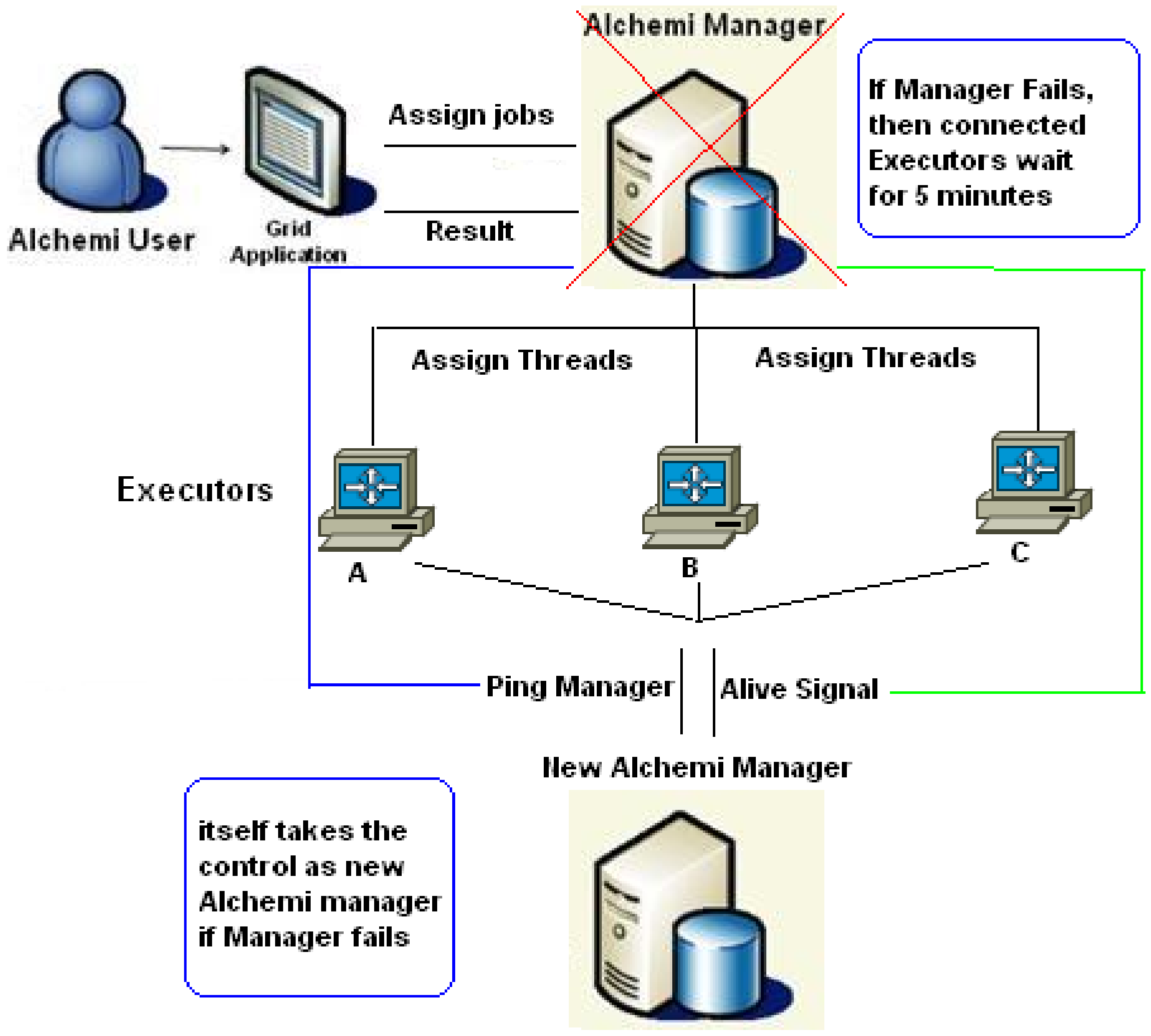


Figure 6.1: Proposed New Manager

6.1.2 Proposed Customized Scheduler Algorithm

The Proposed algorithm described here base on the CPU cycles for solution of executor overload problem of Alchemi grid shown in figure 6.2.

Algorithm 6.1 The Proposed Algorithm

```
1  T is group of threads of Grid Applcation.
2  E is register executors on which the threads are to schedule to execute.
3  Repeat up to all threads from T are scheduled
4  select the threads from group of threads T.
5  select the highest priority threads from the selected thread. Suppose the selected thread is
   Thread1.
6  Select the executor which satisfies the criteria.
7  Criteria
     a. Available CPU cycles=Max CPU cycles * CPU Availability.
8  schedule Thread1 to Executor1 to the Manager.
9  The manager then migrate Thread1 to ExecutorA for the execution.
10 End repeat.
```

Advantage

The algorithm is thread based and check criteria for executor so improve performance of grid.

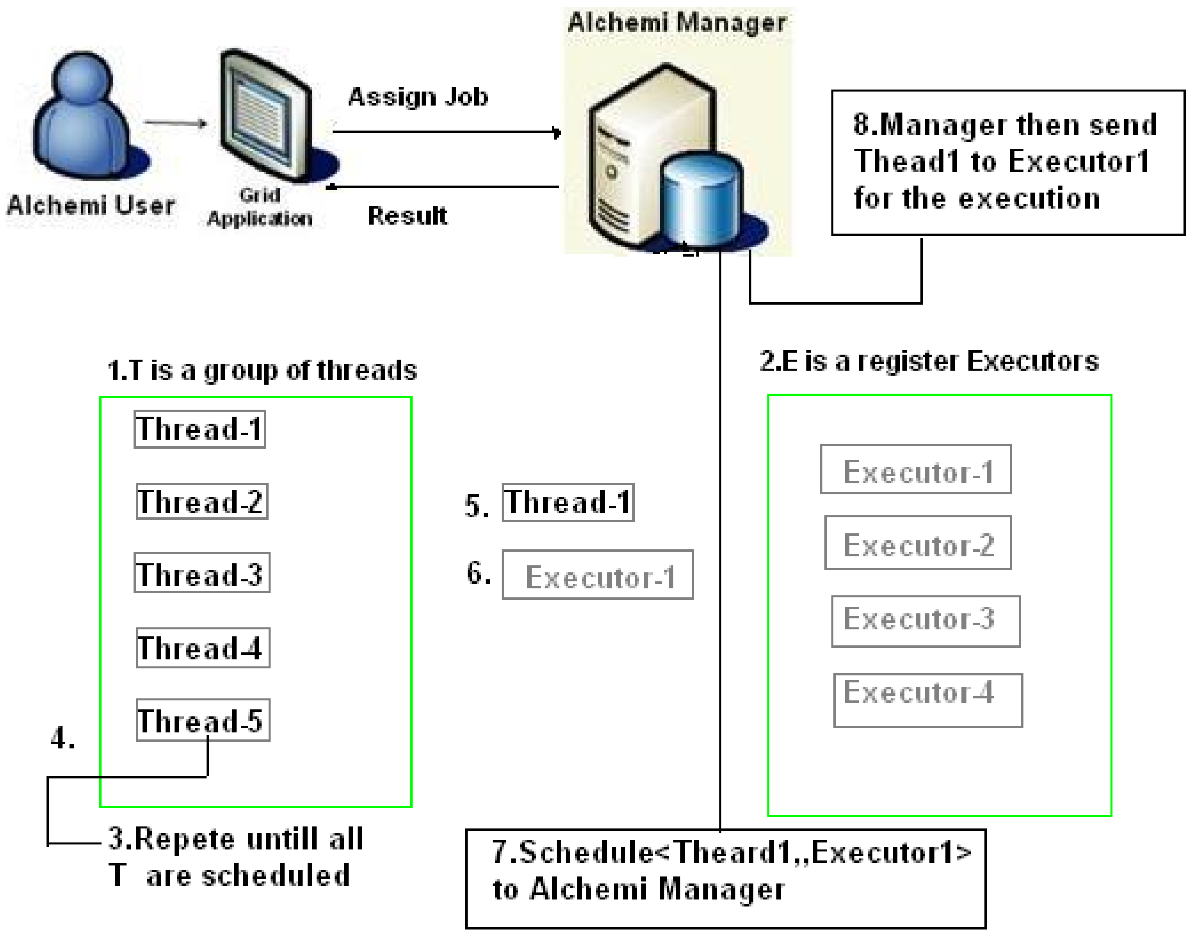

Alchemi Manager
Alchemi User
Grid Application
Assign Job
Result
8.Manager then send Thead1 to Executor1 for the execution
1.T is a group of threads
Thread-1
Thread-2
Thread-3
Thread-4
Thread-5
2.E is a register Executors
Executor-1
Executor-2
Executor-3
Executor-4
5. Thread-1
6. Executor-1
4.
3.Repete untill all T are scheduled
7.Schedule<Theard1,,Executor1> to Alchemi Manager

Figure 6.2: Proposed Customized Scheduler Algorithm

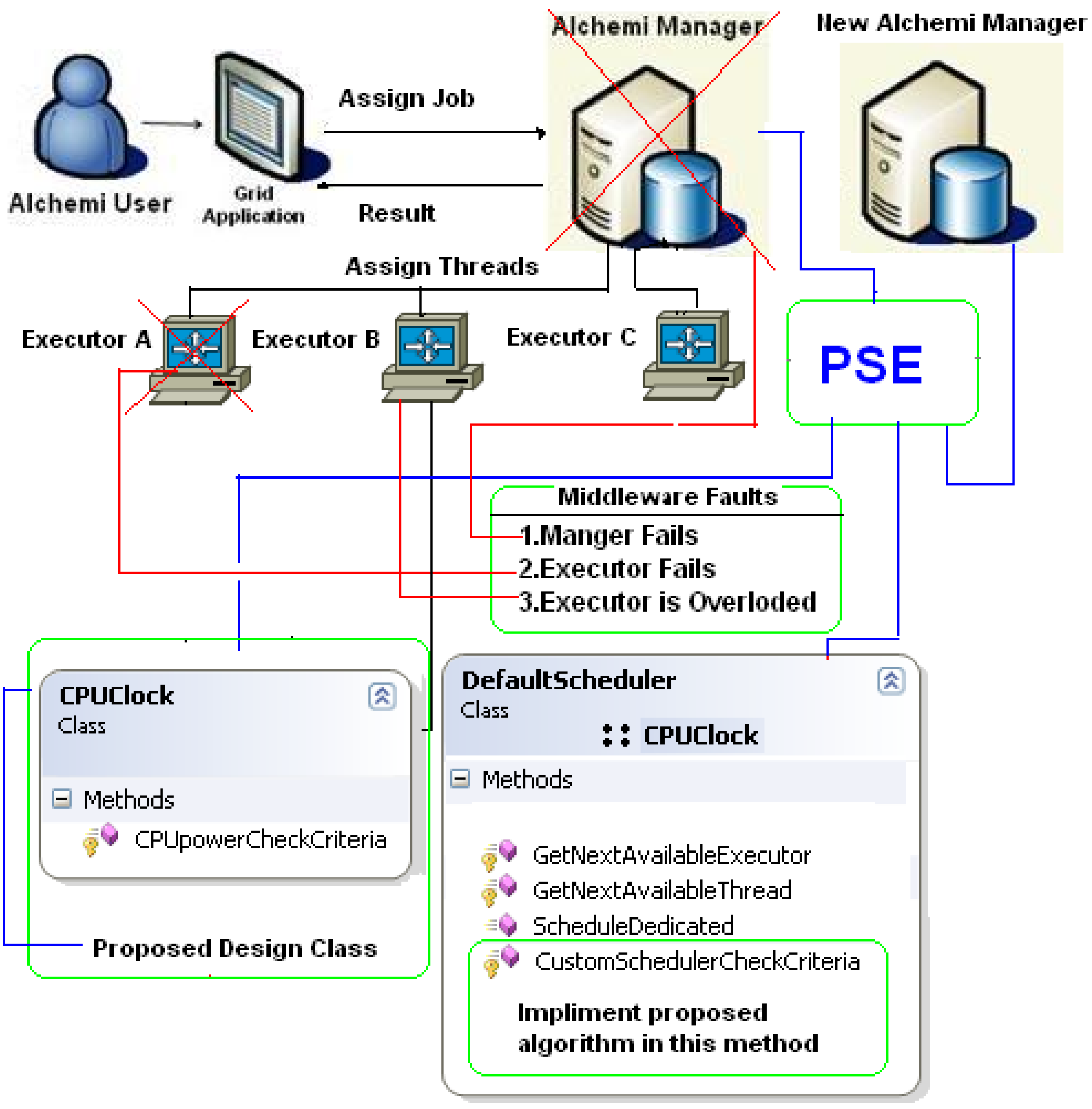


Figure 6.3: Design of Problem Solving Environment

Chapter 7

Implementation

7.1 Alchemi Grid API Main Classes Description

Typically,a user application instantiates 3 classes of objects from the Alchemi API[8]: GConnection, GApplication and one or more GThreads shown in figure 7.1. GConnection is a class responsible for connecting and sending authentication information from the user application to the central component called Manager.The GConnection is consumed by GApplication, which is also created by the user application. To take an advantage of the grid, user application must create a number of independent jobs and submit them to the grid. In Alchemi terminology, the independent jobs are referred to as grid threads. They are objects whose class definition is inherited from GThread class.Alchemi API have Mainly 3 Classes Modules see Table-I,II,III.

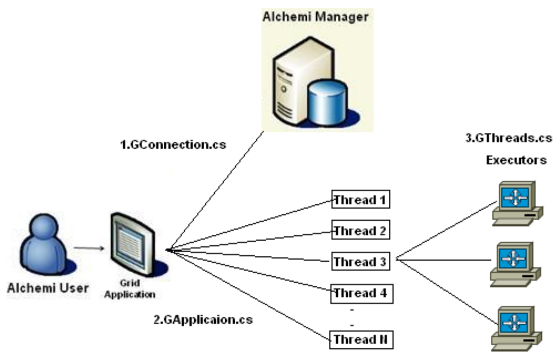


Figure 7.1: Alchemi Grid API main classes

	Alchemi Core Classes Modules
Class Name	**Description**
GThread.cs	This class represent a ”thread” that can be run on a remote grid node.
ThreadCollection.cs	This class represent a collection of GThreads.
ThreadIdentifier.cs	This class represent an identifier to uniquely identify a thread across applications.
ThreadState.cs	This class represent a List of possible thread states.
GNode.cs	This class contains methods for connection between making the node as grid node and relates it with the grid manager.
SecurityCredentials.cs	This class represent the credentials required to authenticate to a node.

Table I: Alchemi Core Classes Modules

	Alchemi Manager Classes Modules
Class Name	**Description**
GManager.cs	This class represents an Alchemi Manager.
MApplication.cs	This class represents an Application on the manager.
MExecutor.cs	This class represents a container for the executor reference held by the manager.
MThread.cs	This class represents a thread on the manager node.
DefaultScheduler.cs	This class represents Scheduling of threads ,works on the basis of priority based FIFO.
Configuration.cs	This class stores the configuration information for the Alchemi Manager includes information such as database details, own port.

Table II: Alchemi Manager Classes Modules

	Alchemi Executor Classes Modules
Class Name	**Description**
ExecutorInfo.cs	This class represents the static attributes of an executor.
GExecutor.cs	The GExecutor class is an implementation of the IExecutor interface and represents an Executor node.
HeartbeatInfo.cs	This structure is a container for all the information sent in a hereatbeat update.
Configuration.cs	This class stores the configuration information for the Alchemi Executor includes information such as the manager host and port no.

Table III: Alchemi Executor Classes Modules

7.1.1 Proposed Algorithm Implementation

- For the implementation[18][21][22] of proposed customized scheduler algorithm for executor overloaded fault proposed one class CPUClock.cs to check criteria and default-scheduler.cs class within Manager Module was modified.Updated Logic in Alchemi Middleware API(Application Programming Interface) Shown in figure 7.2.

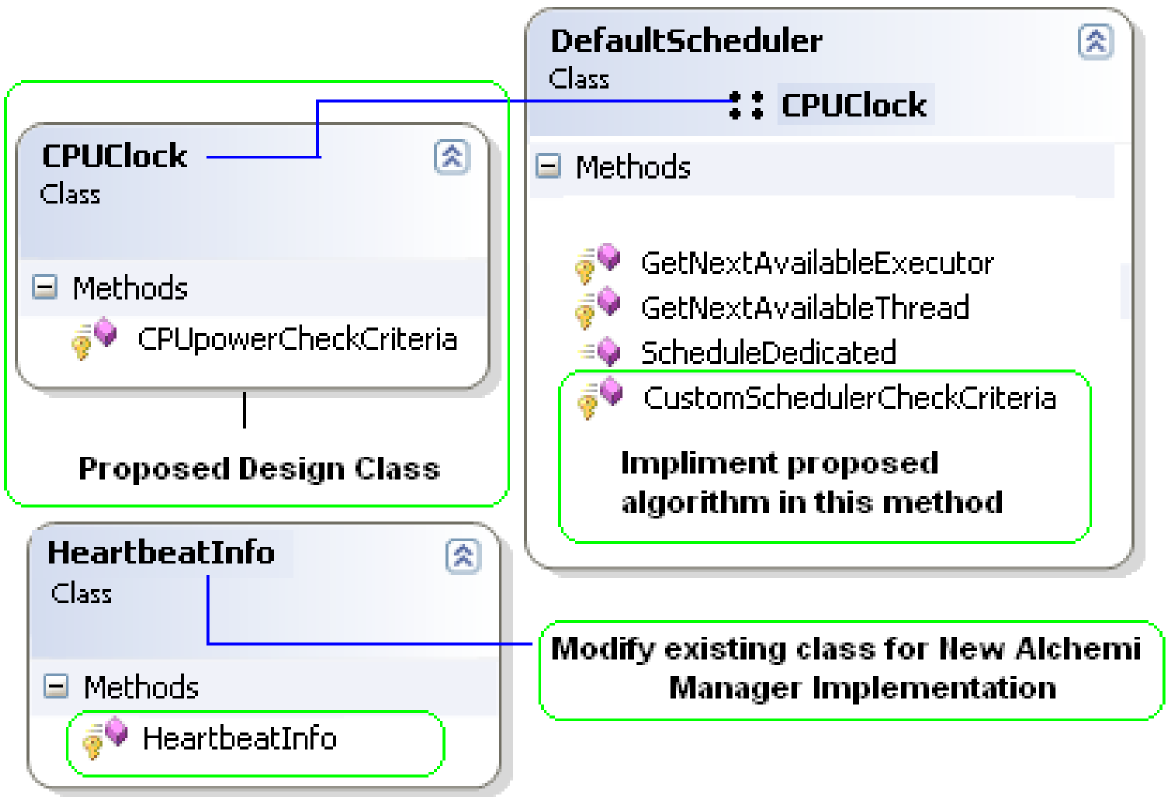


Figure 7.2: Updated Logic in Alchemi Middleware API

7.1.2 Proposed New Alchemi Manager Implementation

- For the implementation[18][21][22] of proposed New Alchemi Manager modify Heartbeatinfo.cs class within Manager Module was modified.Updated Logic in Alchemi Middleware API(Application Programming Interface) Shown in figure 7.2.

Chapter 8

Result

8.1 Alchemi based Grid Application Result

8.1.1 Proposed New Alchemi Manager Result

Functionality of new manager was tested on a job which was developed and deployed on Alchemi Grid, and the results obtained were compared against the existing manager of grid.The proposed New Alchemi Manager was deployed on registered executor takes control on grid after failure of existing manager. Reasons of failure of existing manager could be because of failure in hardware or faults in the network. Results are shown in Table I and figure 8.5,8.6 and 8.7.

8.1.2 Proposed Algorithm Result

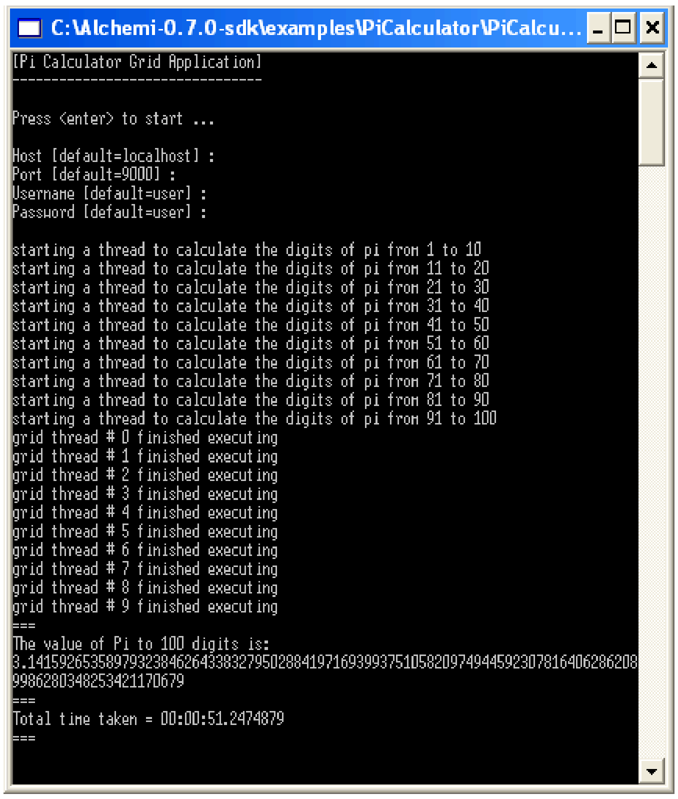
To setup Alchemi grid, first start Alchemi manager node shown in figure 8.2, Alchemi manager is configured to run on the node having IP 10.1.3.16, and there are five Alchemi Executor nodes registerd with the Alchemi Manager as shown in figure 8.3 and 8.8.In this Alchemi Grid, one of the grid application was run and tested for multiple times with different grid configuration. The results obtained from Alchemi grid are compared with the results obtained from Proposed Algorithm which are shown in Table II and figure 8.9,8.10,8.11 and 8.12.

Grid Application (Number of Threads)	Failure of Threads, if manager node fails	% Failure in Alchemi	Proposed New manager Result
10	3	30%	Successfully executed
20	8	40%	Successfully executed
30	14	46%	Successfully executed
40	22	55%	Successfully executed

Table I: Proposed New manager result of Grid Application

Grid Application (Threads,Executors)	Execution Time of Alchemi	Proposed Algorithm Result
(10, 2)	51.2 second	46.5 second
(10, 3)	40.3 second	36.6 second
(10, 4)	31.3 second	29.4 second
(10, 5)	25.4 second	22.1 second

Table II: Proposed Algorithm result of Grid Application



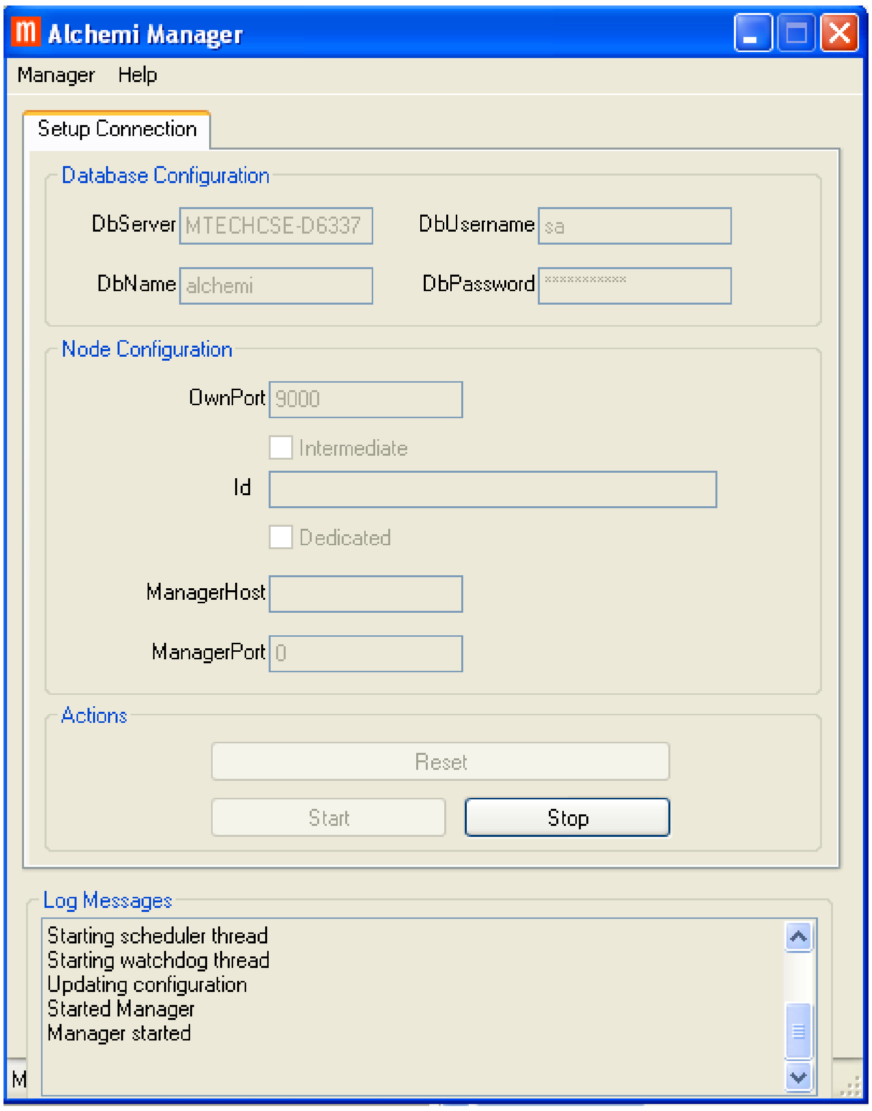
```
[Pi Calculator Grid Application]
--------------------------------

Press <enter> to start ...

Host [default=localhost] :
Port [default=9000] :
Username [default=user] :
Password [default=user] :

starting a thread to calculate the digits of pi from 1 to 10
starting a thread to calculate the digits of pi from 11 to 20
starting a thread to calculate the digits of pi from 21 to 30
starting a thread to calculate the digits of pi from 31 to 40
starting a thread to calculate the digits of pi from 41 to 50
starting a thread to calculate the digits of pi from 51 to 60
starting a thread to calculate the digits of pi from 61 to 70
starting a thread to calculate the digits of pi from 71 to 80
starting a thread to calculate the digits of pi from 81 to 90
starting a thread to calculate the digits of pi from 91 to 100
grid thread # 0 finished executing
grid thread # 1 finished executing
grid thread # 2 finished executing
grid thread # 3 finished executing
grid thread # 4 finished executing
grid thread # 5 finished executing
grid thread # 6 finished executing
grid thread # 7 finished executing
grid thread # 8 finished executing
grid thread # 9 finished executing
===
The value of Pi to 100 digits is:
3.141592653589793238462643383279502884197169399375105820974944592307816406286208
9986280348253421170679
===
Total time taken = 00:00:51.2474879
===
```

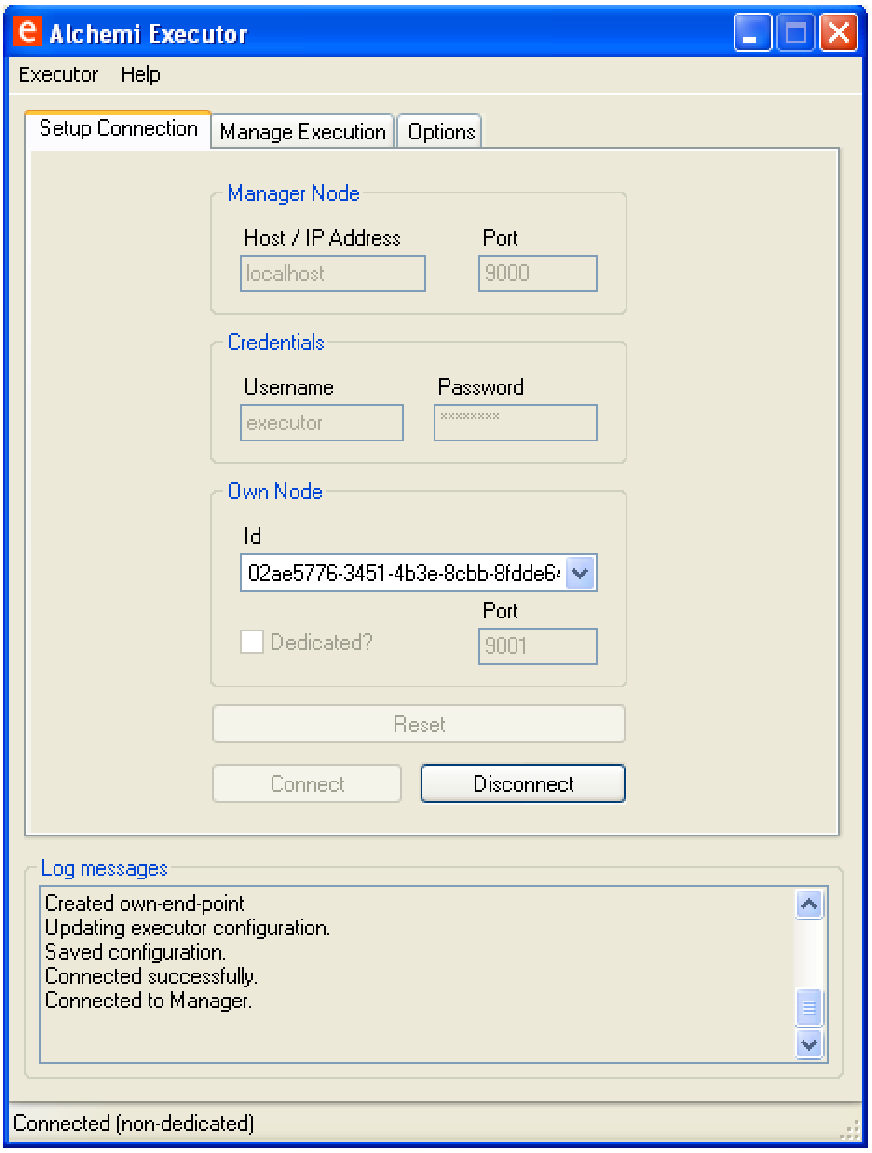
Figure 8.1: Execution of Grid Application using Alchemi



DbServer: MTECHCSE-D6337
DbUsername: sa
DbName: alchemi
DbPassword: ************
OwnPort: 9000
Intermediate
Id:
Dedicated
ManagerHost:
ManagerPort: 0

```
Starting scheduler thread
Starting watchdog thread
Updating configuration
Started Manager
Manager started
```

Figure 8.2: Alchemi Manager starting state

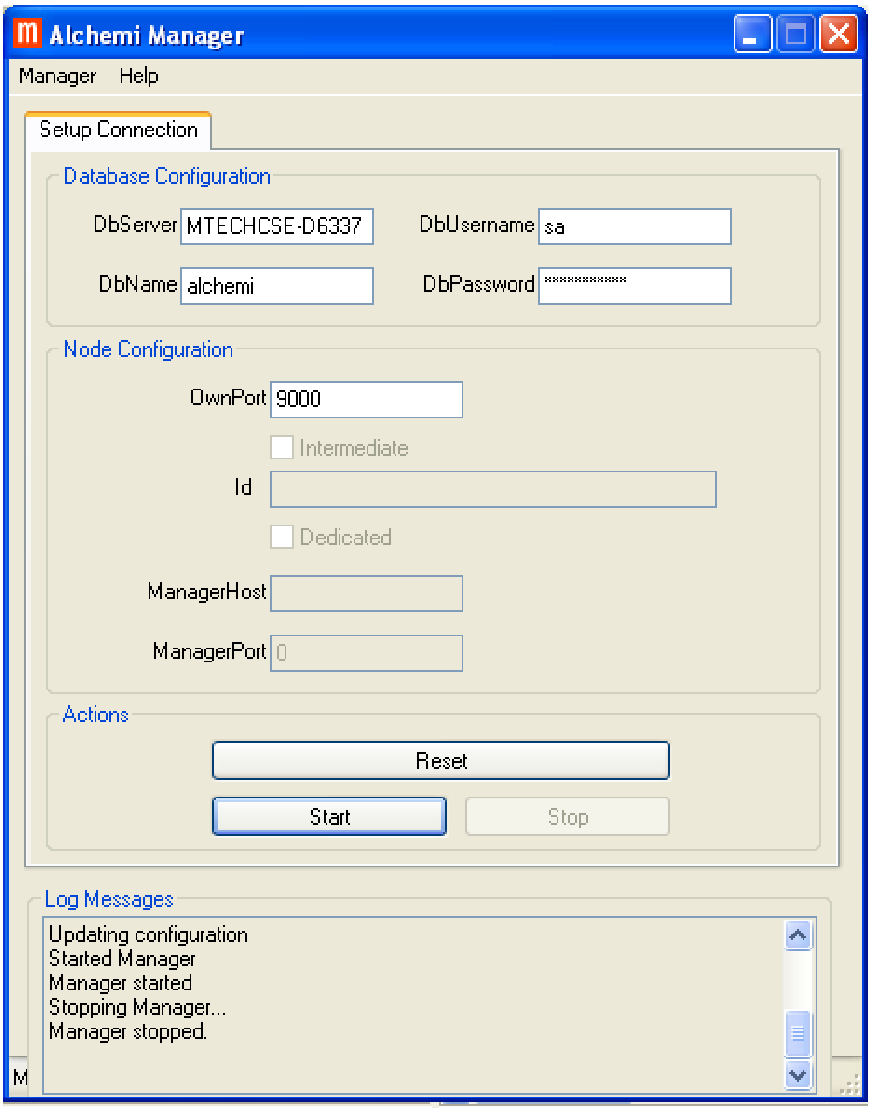


Host / IP Address: localhost
Port: 9000
Username: executor
Password: ********
Id: 02ae5776-3451-4b3e-8cbb-8fdde6
Port: 9001
Dedicated? [ ]

```
Created own-end-point
Updating executor configuration.
Saved configuration.
Connected successfully.
Connected to Manager.
```

Connected (non-dedicated)

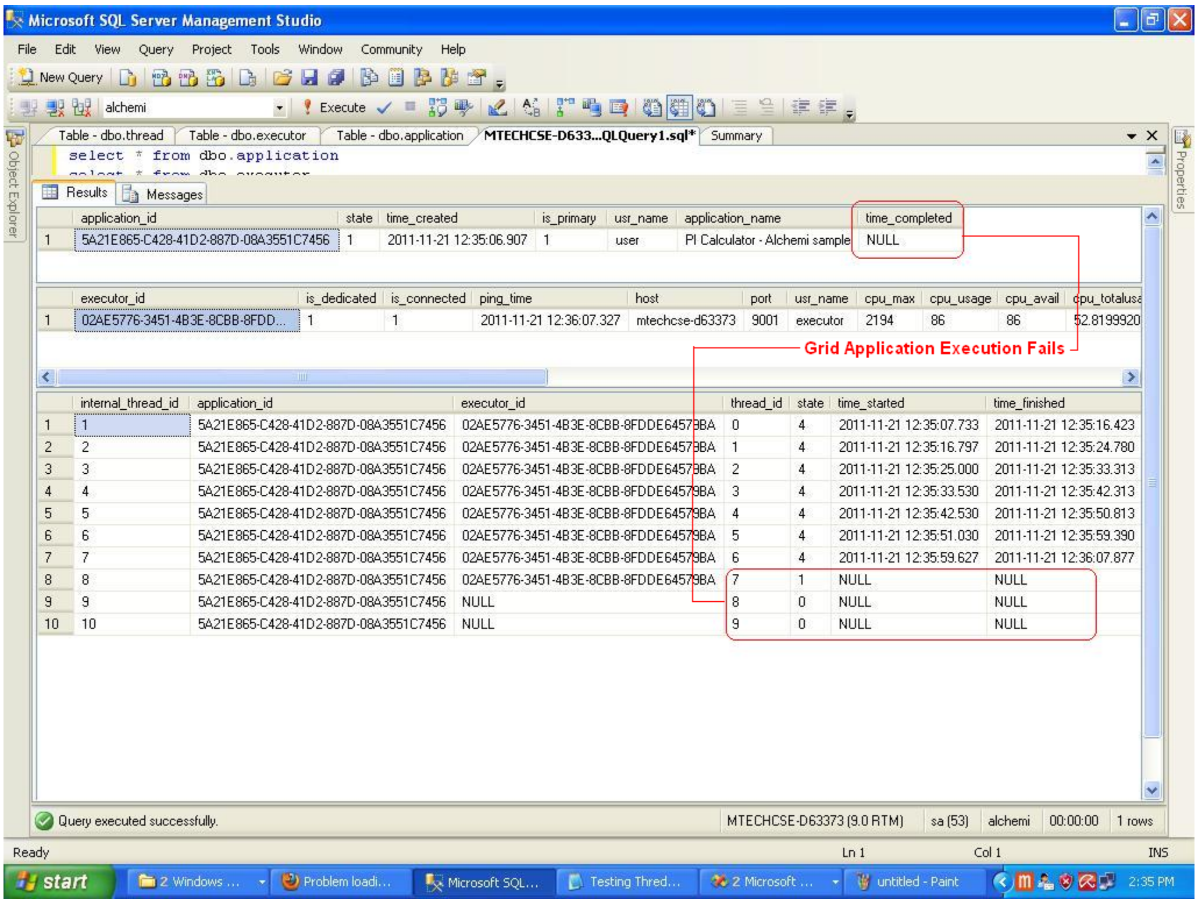
Figure 8.3: Alchemi Executor starting state



DbServer MTECHCSE-D6337
DbUsername sa
DbName alchemi
DbPassword ************
OwnPort 9000
[ ] Intermediate
Id
[ ] Dedicated
ManagerHost
ManagerPort 0

```
Updating configuration
Started Manager
Manager started
Stopping Manager...
Manager stopped.
```

Figure 8.4: Alchemi Manager stoping state



```
select * from dbo.application
```

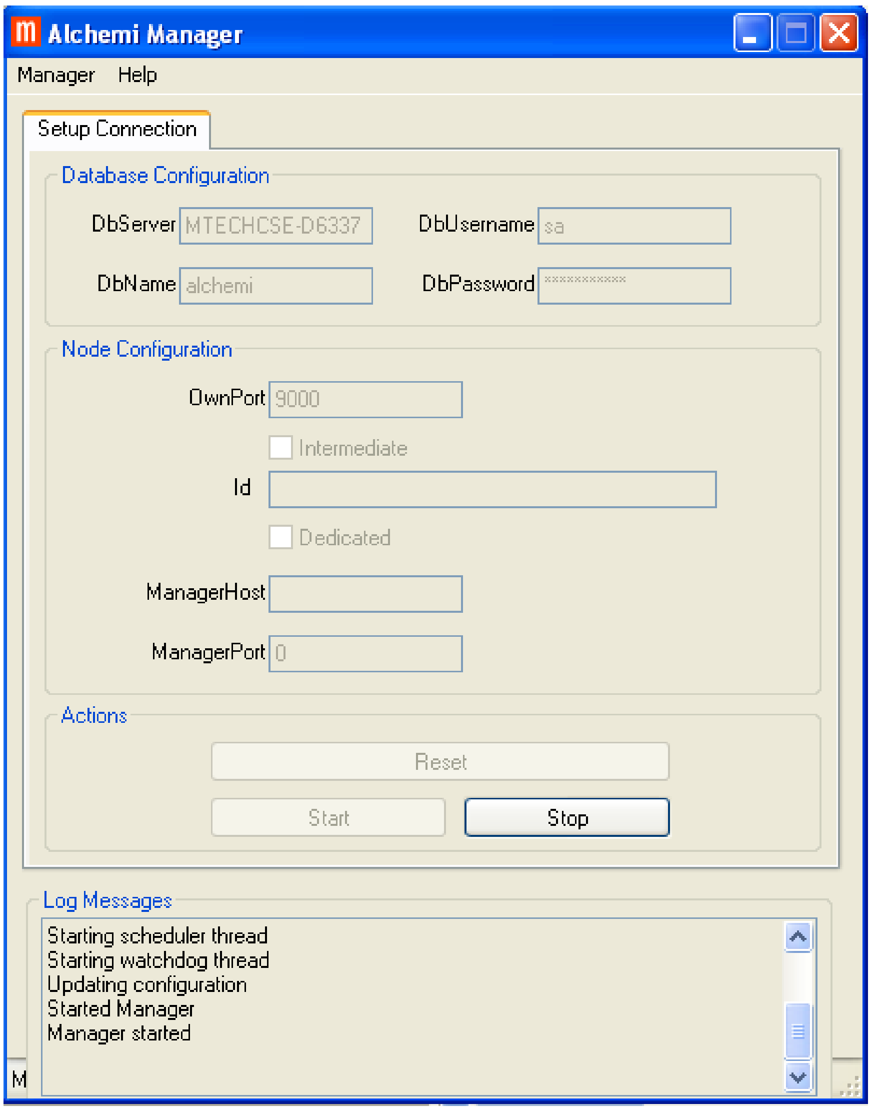
	application_id	state	time_created	is_primary	usr_name	application_name	time_completed
1	5A21E865-C428-41D2-887D-08A3551C7456	1	2011-11-21 12:35:06.907	1	user	PI Calculator - Alchemi sample	NULL

	executor_id	is_dedicated	is_connected	ping_time	host	port	usr_name	cpu_max	cpu_usage	cpu_avail	cpu_totalusa
1	02AE5776-3451-4B3E-8CBB-8FDD...	1	1	2011-11-21 12:36:07.327	mtechcse-d63373	9001	executor	2194	86	86	52.8199920

	internal_thread_id	application_id	executor_id	thread_id	state	time_started	time_finished
1	1	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	0	4	2011-11-21 12:35:07.733	2011-11-21 12:35:16.423
2	2	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	1	4	2011-11-21 12:35:16.797	2011-11-21 12:35:24.780
3	3	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	2	4	2011-11-21 12:35:25.000	2011-11-21 12:35:33.313
4	4	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	3	4	2011-11-21 12:35:33.530	2011-11-21 12:35:42.313
5	5	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	4	4	2011-11-21 12:35:42.530	2011-11-21 12:35:50.813
6	6	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	5	4	2011-11-21 12:35:51.030	2011-11-21 12:35:59.390
7	7	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	6	4	2011-11-21 12:35:59.627	2011-11-21 12:36:07.877
8	8	5A21E865-C428-41D2-887D-08A3551C7456	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	7	1	NULL	NULL
9	9	5A21E865-C428-41D2-887D-08A3551C7456	NULL	8	0	NULL	NULL
10	10	5A21E865-C428-41D2-887D-08A3551C7456	NULL	9	0	NULL	NULL

Query executed successfully. MTECHCSE-D63373 (9.0 RTM) sa (53) alchemi 00:00:00 1 rows

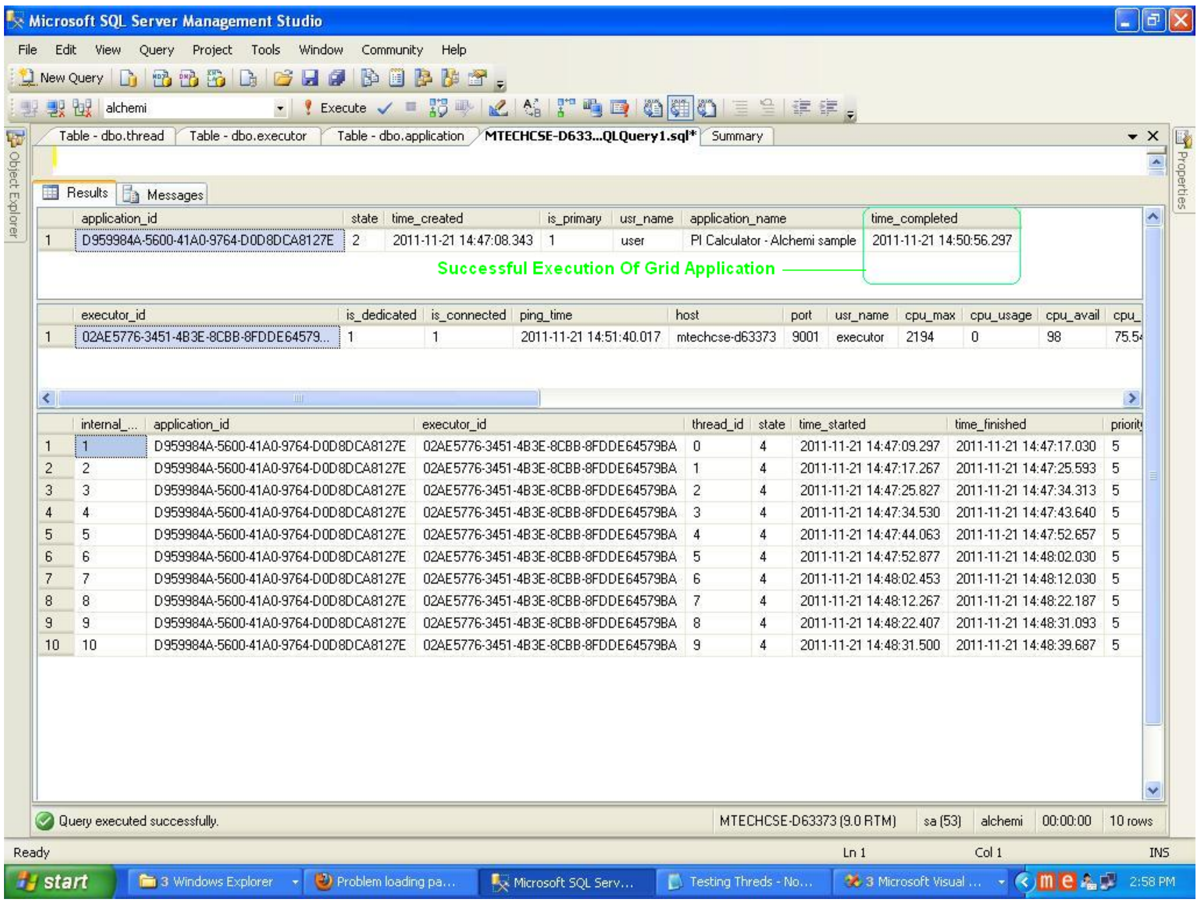
Figure 8.5: If Manager node fails,grid application execution fails



DbServer MTECHCSE-D6337
DbUsername sa
DbName alchemi
DbPassword ************
OwnPort 9000
Intermediate
Id
Dedicated
ManagerHost
ManagerPort 0

```
Starting scheduler thread
Starting watchdog thread
Updating configuration
Started Manager
Manager started
```

Figure 8.6: New Alchemi Manager starting state


Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Community Help
New Query
alchemi
Execute
Table - dbo.thread
Table - dbo.executor
Table - dbo.application
MTECHCSE-D633...QLQuery1.sql*
Summary
Object Explorer
Properties
Results
Messages
Successful Execution Of Grid Application

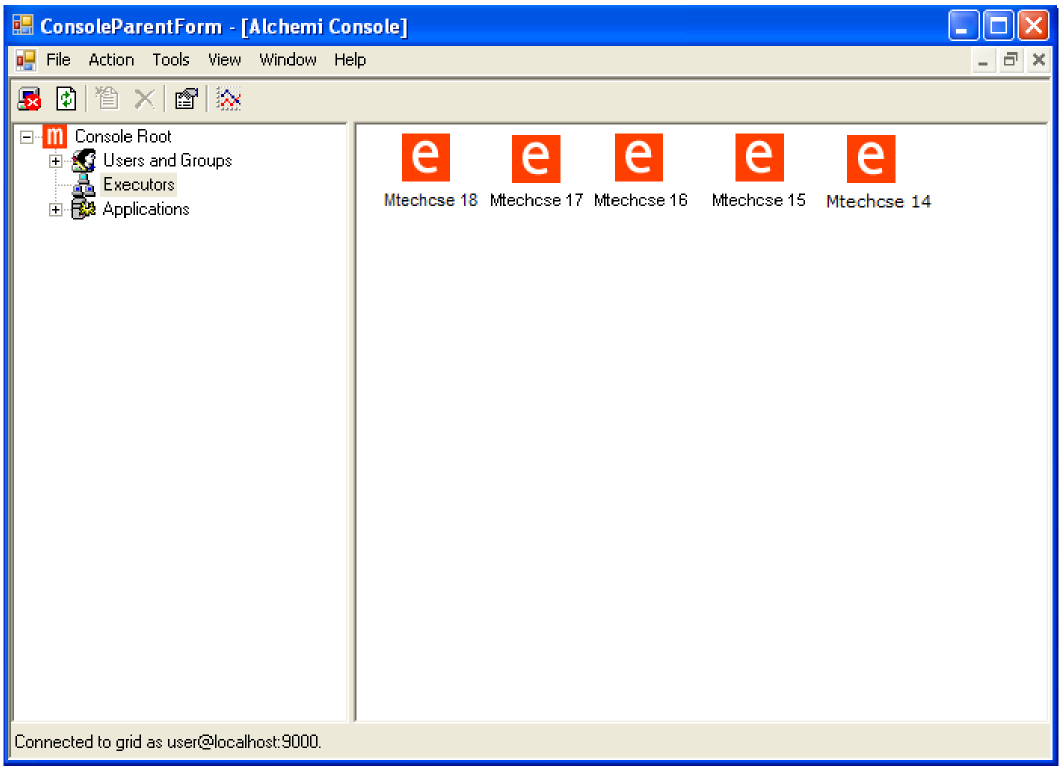
	application_id	state	time_created	is_primary	usr_name	application_name	time_completed
1	D959984A-5600-41A0-9764-D0D8DCA8127E	2	2011-11-21 14:47:08.343	1	user	PI Calculator - Alchemi sample	2011-11-21 14:50:56.297

	executor_id	is_dedicated	is_connected	ping_time	host	port	usr_name	cpu_max	cpu_usage	cpu_avail	cpu_
1	02AE5776-3451-4B3E-8CBB-8FDDE64579...	1	1	2011-11-21 14:51:40.017	mtechcse-d63373	9001	executor	2194	0	98	75.5

	internal_...	application_id	executor_id	thread_id	state	time_started	time_finished	priority
1	1	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	0	4	2011-11-21 14:47:09.297	2011-11-21 14:47:17.030	5
2	2	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	1	4	2011-11-21 14:47:17.267	2011-11-21 14:47:25.593	5
3	3	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	2	4	2011-11-21 14:47:25.827	2011-11-21 14:47:34.313	5
4	4	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	3	4	2011-11-21 14:47:34.530	2011-11-21 14:47:43.640	5
5	5	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	4	4	2011-11-21 14:47:44.063	2011-11-21 14:47:52.657	5
6	6	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	5	4	2011-11-21 14:47:52.877	2011-11-21 14:48:02.030	5
7	7	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	6	4	2011-11-21 14:48:02.453	2011-11-21 14:48:12.030	5
8	8	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	7	4	2011-11-21 14:48:12.267	2011-11-21 14:48:22.187	5
9	9	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	8	4	2011-11-21 14:48:22.407	2011-11-21 14:48:31.093	5
10	10	D959984A-5600-41A0-9764-D0D8DCA8127E	02AE5776-3451-4B3E-8CBB-8FDDE64579BA	9	4	2011-11-21 14:48:31.500	2011-11-21 14:48:39.687	5

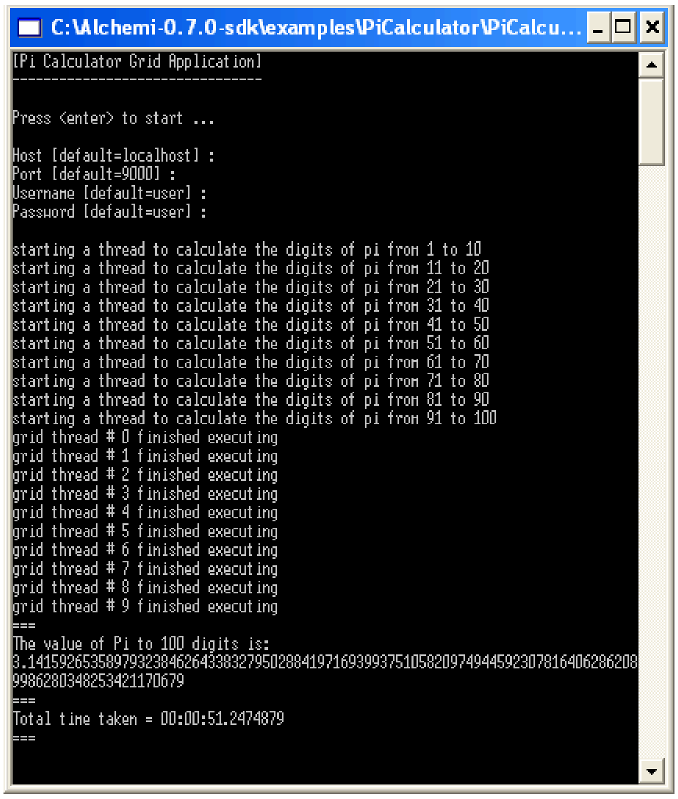
```
Query executed successfully.    MTECHCSE-D63373 (9.0 RTM)   sa (53)   alchemi   00:00:00   10 rows
Ready                                             Ln 1        Col 1        INS
```

Figure 8.7: With New Alchemi Manager,successful execution of grid application



Connected to grid as user@localhost:9000.

Figure 8.8: Grid Console window showing five executor connected to grid manager



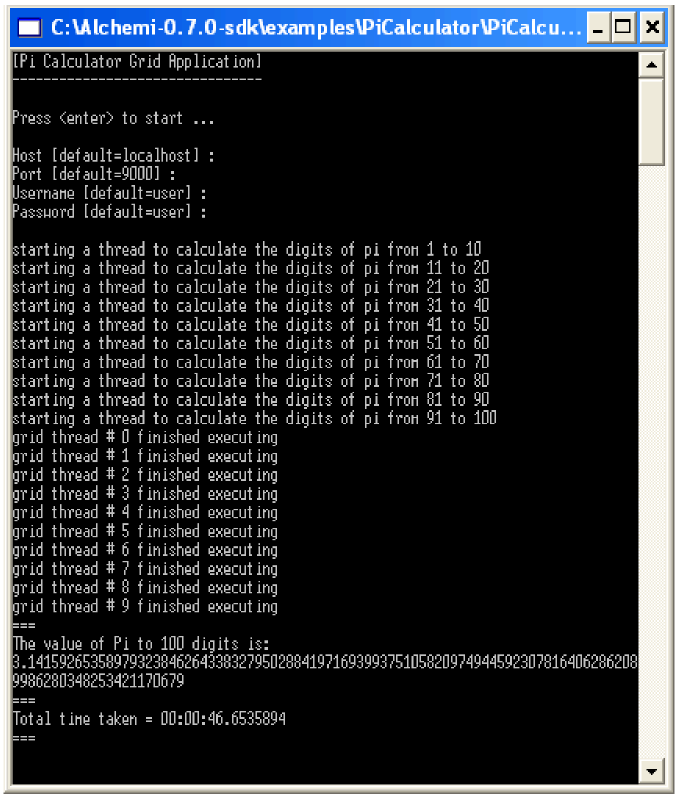
```
[Pi Calculator Grid Application]
--------------------------------

Press <enter> to start ...

Host [default=localhost] :
Port [default=9000] :
Username [default=user] :
Password [default=user] :

starting a thread to calculate the digits of pi from 1 to 10
starting a thread to calculate the digits of pi from 11 to 20
starting a thread to calculate the digits of pi from 21 to 30
starting a thread to calculate the digits of pi from 31 to 40
starting a thread to calculate the digits of pi from 41 to 50
starting a thread to calculate the digits of pi from 51 to 60
starting a thread to calculate the digits of pi from 61 to 70
starting a thread to calculate the digits of pi from 71 to 80
starting a thread to calculate the digits of pi from 81 to 90
starting a thread to calculate the digits of pi from 91 to 100
grid thread # 0 finished executing
grid thread # 1 finished executing
grid thread # 2 finished executing
grid thread # 3 finished executing
grid thread # 4 finished executing
grid thread # 5 finished executing
grid thread # 6 finished executing
grid thread # 7 finished executing
grid thread # 8 finished executing
grid thread # 9 finished executing
===
The value of Pi to 100 digits is:
3.141592653589793238462643383279502884197169399375105820974944592307816406286208
99862803482534211706679
===
Total time taken = 00:00:51.2474879
===
```

Figure 8.9: Execution of Grid Application using Alchemi



```
[Pi Calculator Grid Application]
--------------------------------

Press <enter> to start ...

Host [default=localhost] :
Port [default=9000] :
Username [default=user] :
Password [default=user] :

starting a thread to calculate the digits of pi from 1 to 10
starting a thread to calculate the digits of pi from 11 to 20
starting a thread to calculate the digits of pi from 21 to 30
starting a thread to calculate the digits of pi from 31 to 40
starting a thread to calculate the digits of pi from 41 to 50
starting a thread to calculate the digits of pi from 51 to 60
starting a thread to calculate the digits of pi from 61 to 70
starting a thread to calculate the digits of pi from 71 to 80
starting a thread to calculate the digits of pi from 81 to 90
starting a thread to calculate the digits of pi from 91 to 100
grid thread # 0 finished executing
grid thread # 1 finished executing
grid thread # 2 finished executing
grid thread # 3 finished executing
grid thread # 4 finished executing
grid thread # 5 finished executing
grid thread # 6 finished executing
grid thread # 7 finished executing
grid thread # 8 finished executing
grid thread # 9 finished executing
===
The value of Pi to 100 digits is:
3.14159265358979323846264338327950288419716939937510582097494459230781640628620
89986280348253421170679
===
Total time taken = 00:00:46.6535894
===
```

Figure 8.10: Execution of Grid Application using Proposed Algorithm

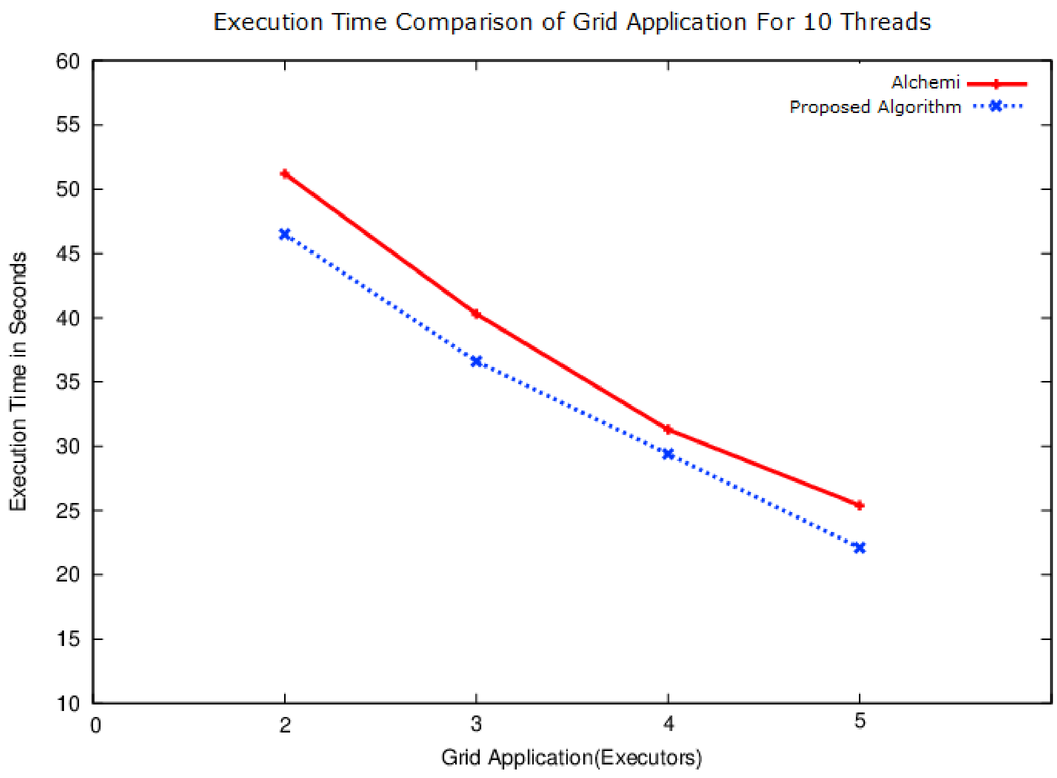


Figure 8.11: Execution Time Comparison line chart

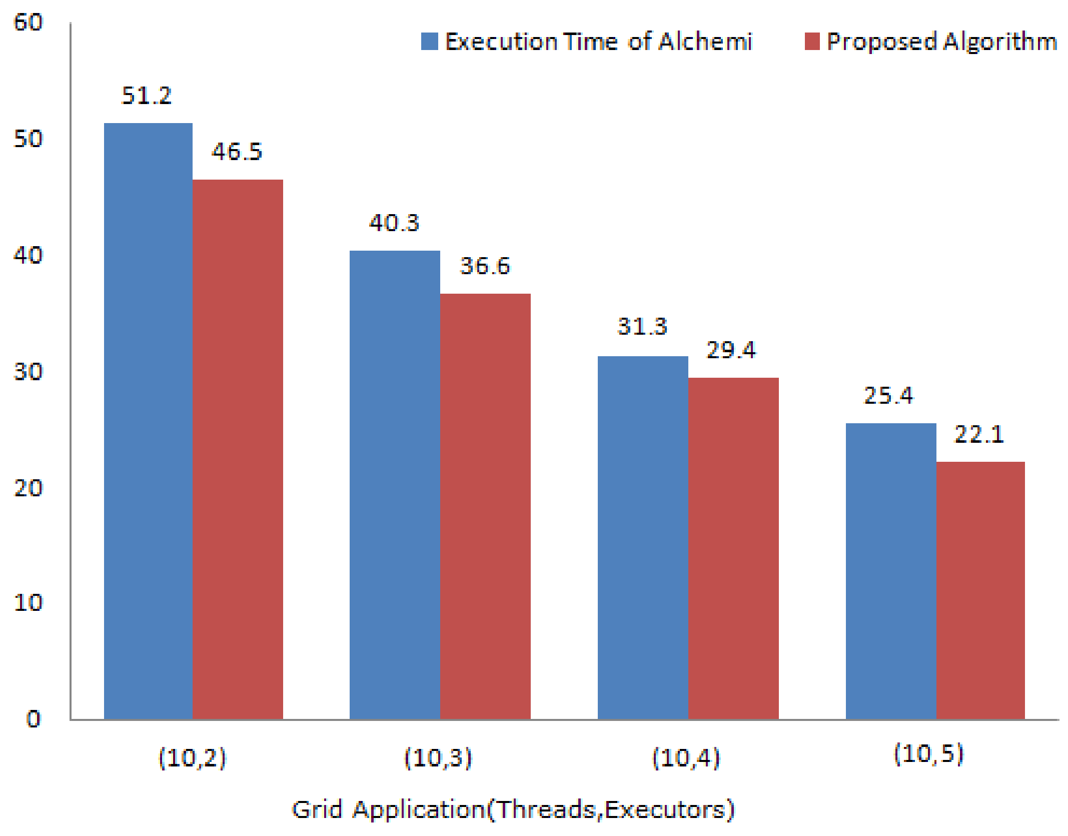


Figure 8.12: Execution Time Comparison bar chart

Chapter 9

Conclusion and Future Work

9.1 Conclusion

- Proposed customized scheduler algorithm and proposed new alchemi manager embedded in problem solving environment and integrated with Alchemi grid so reduced effect of identified middleware faults and grid application execution time is reduced by 6 to 9%.
- Handling middleware faults can lead to an effective and efficient grid environment.

9.2 Future Work

- For the Alchemi manager node fails due to disconnected from the network or hardware failure then all the executor nodes will probably stop their ongoing work. so grid application execution fails solution for such type of fault can be obtained by applying either checkpointing mechanism in which periodical backup of the threads and its states can be taken by the manager or applying failover in which hot backup of the machines are taken. So if any executor fails to finish the execution of the thread then status of execution can be retrieved under checkpointing and can be re-scheduled on another executor for remaining execution continues.

Chapter 10

List of publications

- Yogesh Vaghela,Prof.Madhuri Bhavsar, ***Development of Custom Scheduler To Improve Performance of Alchemi Grid***, at International Conference on Information,Knowledge and Research in Engineering and Technology and Sciences-2012 organized by G.K.Bharad Institute of Engineering,Rajkot and AES,Sangli,Maharastra, India,ISBN No:978-81-906220-3-5,Page 534 to 537 during 24-25 March 2012.
- Yogesh Vaghela, ***Development of Problem Solving Environment in Alchemi Grid***,at National Conference on Advances in Engineering and Technology organized by Kalol institute of Technology and Research Center,Kalol,India during 9-10 March 2012.

References

[1] Rajkumar Buyya, *"High Performance Cluster Computing: Architectures and Systems"*, ISBN 0-13-013784-7, Prentice Hall PTR, NJ, USA, 1999.

[2] Rajkumar Buyya and Srikumar venugopal, *"A Gentle Introduction to Grid Computing and Technologies"*, White Paper,MIELe-Security Pvt. `http://www.buyya.com/papers/GridIntro-CSI2005.pdf`

[3] C.Kesselman. I. Foster, *"Computational grids. In The Grid: Blueprint for a New Computing Infrastructure"*,Morgan-kaufman edition,San Francisco,USA,1999.

[4] Ian Foster, *"What is the Grid? A Three Point Checklist"*, Argonne National Laboratory and University of Chicago.`http://www.gridbus.org/papers/TheGrid.pdf`

[5] S.SivaSathya,K.SyamBabu *"Survey of fault tolerant techniques for grid"*,15740137 @2010 ScienceDirect, Ramanujan School of Computer Science, Pondicherry University, Pondicherry,India.

[6] Raissa Medeiros, Walfredo Cirne, Francisco Brasileiro, Jacques Sauv *"Faults in Grids: Why are they so bad and What can be done about it?"*,0-7695-2026-X/03,@2003 IEEE,Proceedings of the Fourth International Workshop on Grid Computing Universidade Federal de Campina Grande,Paraba,Brazil.

[7] Amit Jain and R.K. Shyamasundar *"Failure Detection and Membership Management in Grid Environments"*,1550-5510/04 @IEEE 2004,Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID04),School of Technology and Computer Science Tata Institute of Fundamental Research Mumbai-400005,India.

[8] Zeljko Stanfel, Goran Martinovic, Zeljko Hocenski *"Scheduling Algorithms for Dedicated Nodes in Alchemi Grid"*,1-4244-2384-2/08,@2008 IEEE,Faculty of Electrical Engineering J.J. Strossmayer University of Osijek,Osijek, Croatia.

[9] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal *"Alchemi.NET Framework in Grid Computing"*,Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering The University of Melbourne, Australia 2003.

[10] Sabir Ismail,Abu Fazal Md Shumon,Md Ruhul Amin *"Distributed Memory Caching for the Fail Safe Computation to Improve the Grid Performance"*,978-1-4244-8494-2/10 @2010 IEEE,Proceedings of 13th International Conference on Computer and Information Technology (ICCIT 2010) 23-25 December, 2010, Dhaka, Bangladesh.

[11] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal *"Peer-to-peer grid computing and a .NET-based Alchemi framework"*,in: HighPerformance Computing: Paradigm and Infrastructure, L. Yang, and M.Guo (Eds.) Wiley Press, New Jersey, USA, 2005.

[12] M.A. Arefin, M.S. Sadik, S. Coetzee, and J. Bishop *"Alchemi vs. Globus: a performance comparison"*,in: Proc. Int. Conf. Electrical and Computer Engineering, pp. 602-605, 2006.

[13] Md. Abu Naser Bikas, AltafHussain, Abu Awal Md. Shoeb,Md. Khalad Hasan, and Md. Forhad Rabbi *"File Based GRID Thread Implementation in the .NET-based Alchemi Framework"*, Multitopic ConferenceI, NMIC. IEEE Intern, pp. 468-472, 2008.

[14] `http://www.gridcomputing.com/`

[15] `http://www.alchemi.net/`

[16] `http://www.cloudbus.org/alchemi/`

[17] `http://www.mail-archive.com/alchemi-developers@lists.sourceforge.net/`

[18] `http://msdn.microsoft.com/en-us/library/`

[19] `http://www.globus.org/`

[20] `http://web.njit.edu/all_topics/HPC/sge.html`

[21] `http://www.codeproject.com/`

[22] `http://csharp-station.com/Tutorial.aspx/`

Index